THE UNIVERSITY OF ALBERTA

CONTROL SYSTEM ANALYSIS PROGRAM

by

(C) WALTER AGOSTINIS

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

IN CHEMICAL ENGINEERING

DEPARTMENT OF CHEMICAL AND PETROLEUM ENGINEERING

EDMONTON, ALBERTA

FALL, 1969

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Control System Analysis Program", submitted by Walter Agostinis in partial fulfilment of the requirements for the degree of Master of Science.

ABSTRACT

This thesis describes a computer program, written for an IBM 1800 data acquisition and control system computer, that will assist with the design and analysis of linear, time-invariant control systems.

Multivariable systems involving pure time delay elements and characteristic equations with repeated and/or complex roots can be handled.

The program will accept as input data a coded equivalent of the classical block diagram plus transfer functions definition of the problem and produce the equivalent matrix formulation of the system state equation. By doing so, the program links the classical transform method of analysis with the more recent state variable approach. The user also has the option of defining and entering his problem directly in the standard state equation form. Regardless of the form of the input, the program calculates the coefficient matrices for the state difference equations and produces the time domain solution for any specified state and/or output variables.

The state transition or fundamental matrix required to derive the state difference representation of a given system is obtained by the Cayley-Hamilton technique. This technique has been extended to handle complex eigenvalues

occurring simultaneously with distinct and repeated ones.

The program has user's intervention capabilities in a "conversational" mode analogous to most simulation programs. It allows the user to rerun a problem having altered, for example, the control parameters or process conditions. The program produces complete problem documentation including plots of the time domain response(s) on either a digital plotter or a storage oscilloscope.

The six examples presented and the use by other students confirm the usefulness of the program to solve realistic control problems.

The control oriented block diagram, transfer function input feature, simplified to the extent of requiring no block diagram manipulation and minimum input data, should encourage process control students to use the program to carry out simulation studies and to gain experience with control techniques.

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to Dr. D. G. Fisher for his supervision of this work.

The staff of the DACS Centre is thanked for their valuable assistance during the implementation of the program.

Financial support from the National Research Council of Canada, grant A-3512 is gratefully acknowledged.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter I

INTRODUCTION

This thesis describes a digital computer program which will facilitate the design, analysis and optimization of of linear control problems.

Development of mathematical models and evaluation of their time domain solutions for different control schemes are basic operations for most control problems.

Fortunately many physical systems can be adequately represented by ordinary differential equations of the type shown in equation (1-1).

$$a_n\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1} y}{dt^{n-1}} + .. + a_1\frac{dy}{dt} + a_0 = f(t) \qquad (1\text{-}1)$$

In the classical method of control system analysis it is assumed that the system represented by equation (1-1) starts initially from a steady state such that all the initial conditions are zero. This assumption simplifies the Laplace Transformation of equation (1-1) so that it can be represented in the "transfer function" form shown in equation (1-2)

$$y(s) = G(s)f(s)$$

$$\text{where } G(s) = \frac{1}{a_n s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_o}$$

$$y(s) = L\left[y(t)\right] \qquad (1\text{-}2)$$

$$f(s) = L\left[f(t)\right]$$

Many control system analysis procedures further assume that the forcing function f(t) is a standard function such as a step, ramp or sine wave. Therefore the analysis, or inverse transform of equation (1-2) is simplified and the properties of the time domain solution y(t) are usually easily determined.

When several components, such as the process, controller, measuring element and control valve are included in the system, then, standard practice is to treat each component separately and develop transfer function relationships similar to equation (1-2). The interrelationship between the components is then represented by a block diagram such as that shown in Figure 1-1.

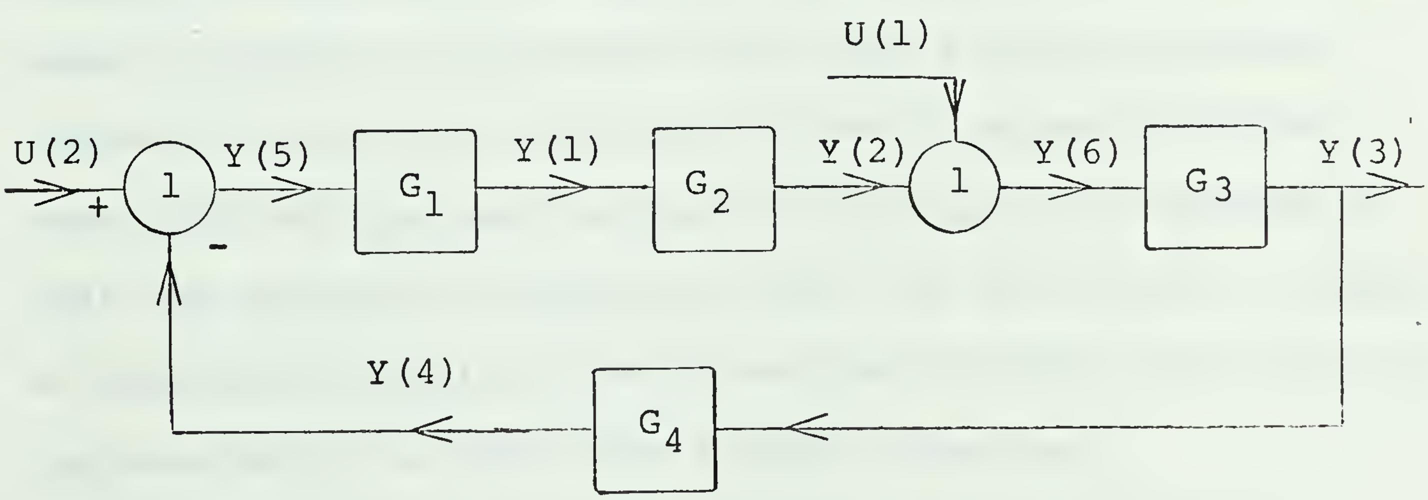


Figure 1-1. Block Diagram for Standard, Single Variable Control System.

The control engineer can then use the familiar techniques of block diagram manipulation and inverse Laplace transformation to determine the system response, Y(3), to load disturbance U(1).

In recent years there has been increasing emphasis on the use of matrix or state variable methods. In this method models are either formulated directly, or converted from forms such as equation (1-1), into a system of first order differential equations which can be represented in matrix notation as

$$\dot{\underline{X}}(t) = \underline{\underline{A}}\,\underline{X}(t) + \underline{\underline{B}}\,\underline{U}(t) \tag{1-3}$$

In the more general case control systems, such as the one shown in Figure 1-1 are represented by the following pair of matrix equations

$$\begin{aligned} \dot{\underline{X}}(t) &= \underline{\underline{A}}\,\underline{X}(t) + \underline{\underline{B}}\,\underline{U}(t) \\ \underline{Y}(t) &= \underline{\underline{C}}\,\underline{X}(t) + \underline{\underline{D}}\,\underline{V}(t) \end{aligned} \tag{1-4}$$

Standard mathematical methods exist for obtaining the time domain solution of equation (1-4) and a rapidly growing volume of literature deals with control aspects such as stability and optimum response. One important feature is that the solution of equation (1-4) can be written in terms of difference equations which are particularly suitable for implementation on real-time digital computers.

The object of this work was to develop a digital computer program that would accept the problem statement in either the block diagram form represented by Figure 1-1

or the matrix equation form defined by equation (1-4) and produce the time domain response for all variables of interest. In addition the program was to include features to assist the engineer in his selection of design variables and to operate in a convenient, interactive mode so that changes could be made to the system specifications and their effect evaluated. Some of the specific design specifications for the program were that it should:

a. Be simple and convenient to use but also capable of handling realistic problems and allowing a knowledgable user to tailor his own solution by the use of options.

b. Run in a time sharing mode on a small real-time computer such as the IBM 1800.

c. Generate the equivalent matrix representation for a problem defined in the block diagram notation.

d. Generate the matrix difference equation form of the solution and if requested generate the time domain solution for all elements of $\underline{X}$ and $\underline{Y}$.

e. Provide the linkage for closed-loop optimization of controller constants and for the use of alternative sub-programs for all internal steps such as finding eigenvalues so that the user can substitute alternative methods.

f. Handle control systems defined by characteristic equations which have real, complex and/or repeated roots.

g. Produce complete problem documentation of both the control results and the important steps of the mathematical solution and display the time domain results as listings, graphs produced on a digital plotter or displays on a storage oscilloscope.

h. Accept forcing functions defined mathematically or simply as a list of function values.

i. Not require any block diagram manipulation or "standardization" by the user prior to using the program.

j. Handle "standard" control problems by simplified options.

k. Handle all the conventional controllers including the ideal proportional plus derivative algorithm.

l. Allow the inclusion of pure time delays as isolated elements or as part of other transfer functions.

The ability to handle pure time delays is extremely important because they are so common in mathematical models of process equipment. In addition several people[18],[20] have demonstrated that a transfer function of the form given in equation (1-5) adequately describes most process units for the purpose of control:

$$G(s) = \frac{ke^{-\tau_d s}}{(\tau_1+1)(\tau_2 s+1)} \qquad (1\text{-}5)$$

If a process is defined by some or all of the terms in equation (1-5) then methods are available in the literature

that could be included in the program to generate the optimal constants for a conventional proportional-integral-derivative controller.

The program as finally developed is described at three different levels of detail. The most general description is given by Figure 2-1 and the accompanying program description in Chapter II. Most of the functions and options in the program are illustrated in the two page flow-diagram included in Appendix B. Finally the details, definitions of flags, provision for future additions, etc., are all defined in the fourteen page flow-diagram included in Appendix B. The complete program listings which are included in Appendices E and F contain comment cards which further explain the details of the program.

The literature survey includes some of the more pertinent references dealing with the mathematical methods used in the program and some of the control oriented features that were incorporated.

The mathematical basis of the program and a presentation of some of the less familiar methods used is given in Chapter IV. The operating system and the limited amount of available core storage of the IBM 1800 computer located in the Department of Chemical and Petroleum Engineering significantly effected the design of the program. Comments on the implementation and the system design are collected in Chapter V.

Those readers interested in the control aspects and

applications should be able to proceed from the general program description to the discussion of examples included in Chapter VI.

The final chapter includes a discussion of user experience, recommendations for further work and the conclusions.

Chapter II

PROGRAM DESCRIPTION

Figure 2-1 shows the overall organization of the program. The first section handles the input and storage of coded block diagram, transfer function and configuration data. In the computational section that follows, the transfer function and configuration data are processed into the equivalent matrix state equation and algebraic matrix equation which have the form:

$$\dot{\underline{X}} = \underline{A}\underline{X} + \underline{B}\underline{U} \qquad (2\text{-}1\text{-}1)$$

$$\underline{Y} = \underline{C}\underline{X} + \underline{D}\underline{V} \qquad (2\text{-}1\text{-}2)$$

Upon user's option the coefficient matrices of system (2-1) are printed out.

The program then calculates the coefficient matrices of the following state difference equation form of the solution:

$$\underline{X}_{j+1} = \underline{\phi}(T)\underline{X}_j + \underline{H}(T)\underline{U}_j \qquad (2\text{-}2\text{-}1)$$

$$\underline{Y}_j = \underline{C}\underline{X}_j + \underline{D}\underline{V}_j \qquad (2\text{-}2\text{-}2)$$

where $\underline{X}_j$ represents the discrete value of $\underline{X}$ at time jT. The basic step in the solution is the calculation of the fundamental or state transition matrix, $\underline{\phi}(T)$. The coefficient matrix $\underline{H}(T)$ is then evaluated by numerical

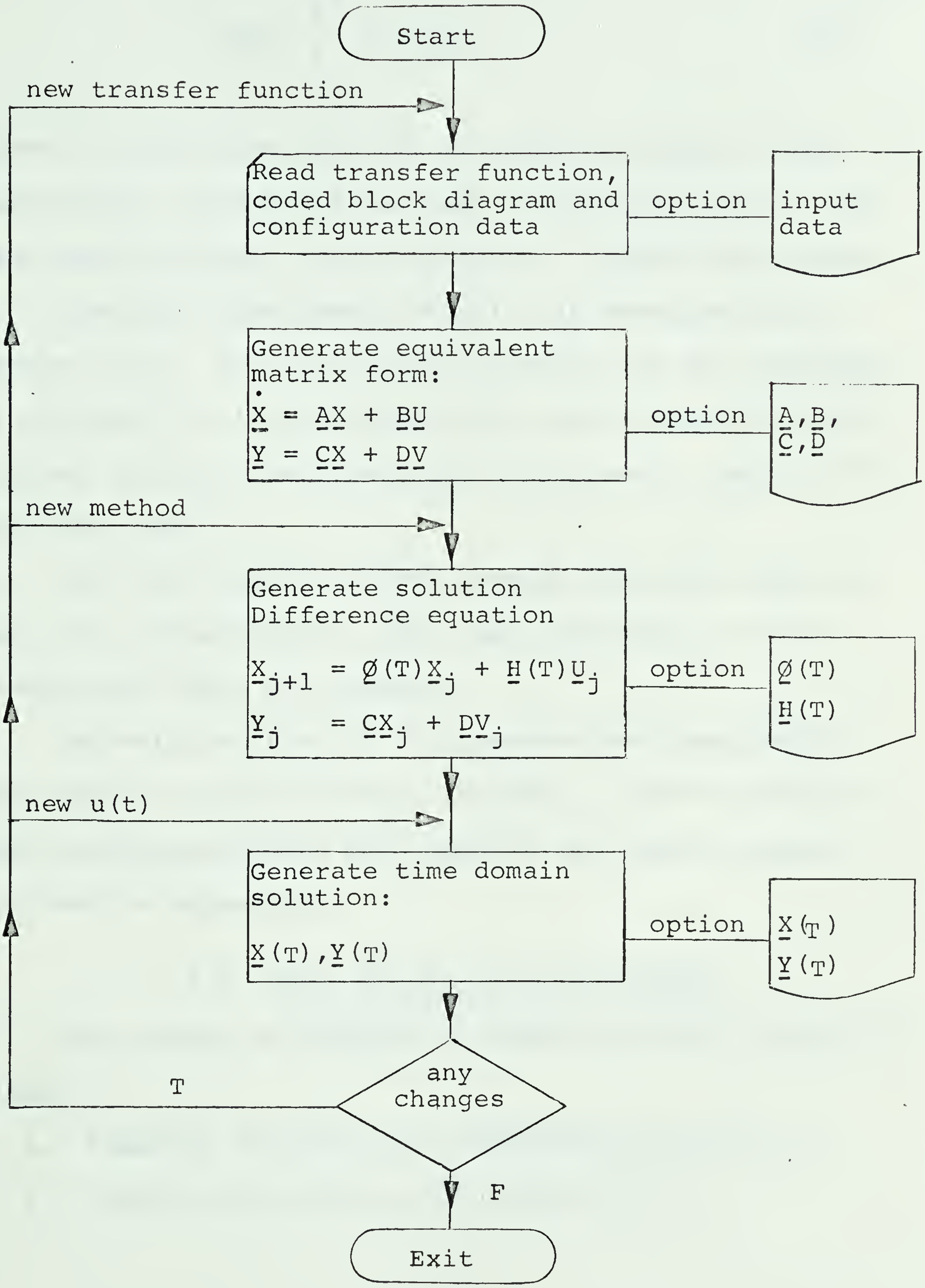


FIGURE 2-1 Simplified Flow Diagram of the Program

evaluation of the following equation:

$$\underline{H}(T) = \int_{0}^{T} \underline{\phi}(t)\underline{B}dt \qquad (2\text{-}3)$$

where T is the time interval or sampling period. The matrices $\underline{B}$, $\underline{C}$ and $\underline{D}$ are the same as defined in (2-1) and, upon user's option, can be printed or punched onto cards.

Finally a time domain solution is generated from system (2-2). The time domain solution for the variables of interest is, in accordance with user's specifications, printed, plotted, or stored into a permanent computer file for later use.

The final section of the program interacts with the user in a conversational mode, and allows him to make changes and rerun his problem.

The main sections of this program are described in more detail in the following sections. Persons wishing to use the program should also consult the "User's Manual" included as Appendix A.

2.1 Input Section of the Program

The program is designed to handle problems defined by either:

a. Transfer function plus configuration data, or

b. Coefficient matrices for equation (2-1).

2.1.1 Transfer Function plus Configuration Data

The classical means of representing a control system is by means of a block diagram and appropriate transfer functions. The interconnections between the various blocks of the diagram are coded for input to the computer program. The required "configuration data" for any particular block consists simply of an identification number, a list of the variables entering the block and as many coefficients as there are inputs. The requirement of entering coefficients as part of the configuration data could have been eliminated by having the user include them in the transfer functions. However an analysis of typical problems showed that the use of separate coefficients permitted greater flexibility and simplicity in defining problems.

Because of the simple and logical type of input the user may enter his transfer function and configuration data almost directly from the block diagram. This is a considerable advantage in comparison with some digital simulation languages (CSMP, LCAP, etc.) that require the user to manipulate the block diagram into a standard form or to transform it into an equivalent system of standard function elements. However it is sometimes possible to significantly reduce the dimensions of equation (2-2-2) by simple manipulation of the block diagram to combine algebraic operations.(*) Examples are discussed in Chapter VI.

For purposes of this program it is assumed that most transfer functions consists of terms that can be

---

(*) see Figure A-1 page A-11.

represented as polynomials in the Laplace transform variable, s. The input data required by the program is the block identification number and the order and list of coefficients for each polynomial term. The method of handling special functions such as time delays and isolated derivative terms is very similar and is explained in the User's Manual, Appendix A.

2.1.2 Matrix Input Option

The emphasis in this case is on using the program as a linear matrix differential equation solver. In most general cases the user must supply the coefficient matrices $\underline{A}$, $\underline{B}$, $\underline{C}$ and $\underline{D}$ of system (2-1) as well as the information necessary to build the forcing function vectors $\underline{U}$ and $\underline{V}$. In the absence of time delays the forcing function vectors $\underline{U}$ and $\underline{V}$ are equal and the input is simplified. Options are included to simplify the handling of special cases of system (2-1) such as the following:

$$\begin{aligned} \dot{\underline{X}} &= \underline{A}\underline{X} + \underline{B}\underline{U} \\ \underline{Y} &= \underline{C}\underline{X} \end{aligned} \tag{2-4}$$

$$\dot{\underline{X}} = \underline{A}\underline{X} + \underline{B}\underline{U} \tag{2-5}$$

Further details on this type of input are given in the User's Manual, Appendix A. The specification of the type and magnitude of external forcing functions is the same for all cases and requires a code number plus one or two parameters.

Auxiliary information such as the initial state vector, the value of the time interval, the desired output variables for the discrete solution, etc. are set to standard values by the program. This is done to allow the user to run a problem with the minimum amount of input data.

It is possible, however, for the user to select his own options and other than "standard" values to fit his specific requirements. Full details are given in Appendix A.

2.2 Generation of the State Equation

Each transfer function is decomposed into an equivalent set of first order linear differential equations by introducing the appropriate "state variables." A matrix representation is adopted. Coefficient matrices of the system of linear differential equations and the associated algebraic equations are manipulated into the standard form of equation (2-1). Refer for details to Chapter IV: Theory and Mathematical Basis.

2.3 Calculation of State Difference Equation

The discrete version of the solution of equation (2-1) is given by equation (2-2). The derivation of this equation is given in Saucedo and Schiring (23).

$$
\begin{aligned}
\underline{X}_{j+1} &= \underline{\phi}(T)\underline{X}_j + \underline{H}(T)\underline{U}_j \\
\underline{Y}_j &= \underline{\underline{C}}\underline{X}_j + \underline{\underline{D}}\underline{V}_j
\end{aligned}
\tag{2-2}
$$

where: T = time interval or sampling period

$\underline{\phi}(T)$ = fundamental or state transition matrix

$$\underline{H}(T) = \int_0^T \underline{\phi}(t)\underline{B}dt$$

$\underline{X}_j$ represents the discrete value of $\underline{X}$ at time jT.

The discrete values of $\underline{X}$ and $\underline{Y}$ generated by equation (2-6) are exact solutions to the matrix equation (2-1) assuming the forcing functions $\underline{U}$ and $\underline{V}$ are constant over each sampling interval. For problems that do not include pure time delays $\underline{U} = \underline{V}$ and the user will usually know from physical considerations whether the assumption is valid. However in the more general case $\underline{U}$ and $\underline{V}$ are partitioned vectors containing forcing functions, delayed forcing functions and delayed state variables (see Chapter IV for details). Consequently, it is more difficult for the user to determine the validity of assuming constant $\underline{U}$ and $\underline{V}$ without examining the structure of these vectors. In general, if $\underline{U}$ and $\underline{V}$ are not constant over the sampling interval, T, then the accuracy of the solution increases as the time interval is reduced.

Clearly, the central objective of this part of the program is the computation of the fundamental matrix. Two methods are presently implemented for this purpose.

a. A power series method based on the definition of the fundamental matrix (see section 4.9).

b. Cayley-Hamilton technique. It is based on the Cayley-Hamilton theorem and uses n-1 power terms of the matrix $\underline{A}$ to find the fundamental matrix where n is the dimension of the square matrix $\underline{A}$.

The Cayley-Hamilton technique as described in the references consulted for this work (8,22,23) was extended to handle complex conjugate eigenvalues occurring alone or in combination with distinct and/or repeated eigenvalues. The details of this extension are given in Chapter IV.

2.4 Time Domain Solution and Output

The time domain solution is generated by iterating equation (2-6) over the desired time range.

Several options are available to display or document the results of the problem analysis. For example the time domain values of $\underline{X}$ and $\underline{Y}$ can be listed on the printer, displayed on an oscilloscope or graphed on the digital plotter. Several levels of documentation of the parameters used and intermediate results are also available.

These features are illustrated by the examples included in Chapter VI: Program Demonstration and Application and in Appendix C. Details on the use of plotting options are included in the User's Manual, Appendix A.

2.5 Changes to Problem Specification

Once the time domain response has been found the program allows the user at the console to make the

following changes:

2.5.1 Disturbance Type and/or Magnitude

This option requires re-execution of only the last part of the program, specifically the time domain solution. Such option allows the user to investigate the effect of different types of forcing functions (step, ramp, sine wave etc.) acting as set points, load disturbances, etc.... on the response of an "output" variable. Minor changes are also allowed. They are illustrated in the User's Manual, Appendix A.

2.5.2 Change of Calculation Pattern

The use of these options basically allow the user to change the method or the numerical techniques used to generate and output the time domain solution. For example:

a. Provision has been made so the user can add a subroutine for direct integration of equation (2-1).

b. The fundamental matrix $\underline{\phi}(T)$ can be generated by either the power series or Cayley-Hamilton techniques.

c. Alternative methods can be used to evaluate eigenvalues.

d. Output can be printed or plotted.

A table of all available options and their code numbers is included in Appendix A.

2.5.3 Transfer Function Parameter(s) and/or Type(s)

This is the most general type of change. It allows the user to change one or more process parameters, types of transfer function, controller parameters, types of controller etc. Because of the program structure, changes described under 2.5.1 and 2.5.2 are also possible at the same time.

As mentioned earlier this change allows the user at the console keyboard to alter one (or more) controller constants based on direct observation of the previous runs. Extensive use of this option was made in providing figures and examples to demonstrate the program included in Chapter VI to which the reader is referred to for more detail.

Chapter III

LITERATURE REVIEW

This program performs a function similar to existing digital simulation programs. Franks (9, p.34) presents a summary of digital simulation languages for continuous systems and reference (3) includes a discussion of the history of some of these programs. Examples are MIMIC and DSL/90 (Digital Simulation Language). CSMP/360 (Continuous System Modeling Program), the most comprehensive of this class, was derived from DSL/90. It is, however, intended for execution on a large scale digital system and is available at the University of Alberta as part of the 360/67 library. Originally designed to overcome some of the limitations of analog simulation as, for example, scaling problems caused by limited voltage levels, patching, etc., they developed into continuous system simulators where the element or block modeling feature of the analog simulation was combined with algebraic and logical modeling capabilities. The input language permits configuration and block statements to be prepared directly and simply from either a block diagram or a differential equation representation of the system to be simulated. Recent improvements

include the implementation of user's intervention capabilities during or at the end of a simulation run in a "conversational" mode as for example in the PACTOLUS language or the University of Alberta version of IBM 1130 CSMP. Because of its analogy with the presented program and its design for implementation on the same system (IBM 1800 DACS computer), the 1130 CSMP will now be described in more detail (11). It is defined as a general "differential equation solver" program. It can handle linear, nonlinear, continuous and discrete systems. The block diagram representation must be manipulated into a series of standard function elements or blocks and, if necessary, user defined ones. Input is given by means of three different types of statements defining block interconnections and functional operations, problem parameters, and external forcing functions. The solution technique consists of feeding the output of an integrator to the next function element in the desired calculation pattern, etc. The final output is given at specified time intervals as a print-plot on the printer or on the digital plotter if one is available.

Discussing the computer implementation of classical control system techniques, Saucedo and Schiring (23) present another program LCAP (Linear Control Analysis Program) which was developed by E. A. Lee, Aerospace Corporation, El Segundo, California. It deals with multiloop, multivariable continuous and discrete systems.

Acceptable inputs are any number of transfer functions with a maximum order of three. However the block diagram, must be manipulated to conform with a "generalized system model" and all external forcing functions must be expressed in Laplace transform before the problem can be entered into the LCAP program. No time delays are allowed. The program then calculates the closed loop transfer function, expands it into "partial fractions" and finds the time domain solution by inverting each term of the expanded expression. The program is also capable of generating, upon request, Bode, Nyquist, Nichols and Root Locus plots.

A paper by Davison (7) presents a similar method for solving large systems of linear differential equations. A linear, first order matrix differential equation is solved by taking the Laplace transforms and expanding the result into partial fractions. An interesting comparison with Runga-Kutta integration shows that the computer time required to solve a problem by direct integration is much longer than that required by the method presented in the paper and for large problems becomes impractical.

Many textbooks deal with the topics in process control of interest to this work (6,8,22,23). DeRusso, Roy and Close (8, p.270) was found particularly helpful for its generalization of the Cayley-Hamilton technique to matrix functions. This is illustrated for the case of distinct and repeated eigenvalues in their examples 4.10.11 and 4.10.12. Saucedo and Schiring (23, p.647) derive the state

difference equation from the state equation and discuss many of the methods incorporated into this work.

Several papers were studied in connection with the implementation of a "library" of functions to generate constants for standard types of process transfer functions. Some of them (1,4,5,20,21,25) refer to analog controllers; others (10,16,17,18,19,24,25) refer to digital controllers. In particular, a paper by Miller, Lopez, Smith and Murrill (20) and one by Chien, Hrones and Reswick (5) provided the data for the proportional-integral-derivative controller constants in a single unity feedback loop with a first order lag plus time delay process transfer function that was used in this program.

A paper by Bakke (2) describing a simplified transient response type of analysis to determine process transfer function parameters was found of interest for its possible implementation to extend the capabilities of the present program. Reference(13)contains the description and listing of all the IBM "Scientific Subroutine Package" subroutines used in the program and not included in the specific Appendix F.

Reference was also made to the series of IBM publications that describe the IBM 1800 computer system and are available in the Department of Chemical and Petroleum Engineering. Of particular use in the programing phase of this project were the FORTRAN IV manual (14) and the Time-Sharing Executive System Operating Procedures (15).

Chapter IV

THEORY AND MATHEMATICAL BASIS

A block diagram representation of a linear system consists of several blocks such as the one shown in Figure 4-1. The multiple inputs are related to the single output through the transfer function, $G_i(s)$ which is normally an algebraic expression in the Laplace variable "s".

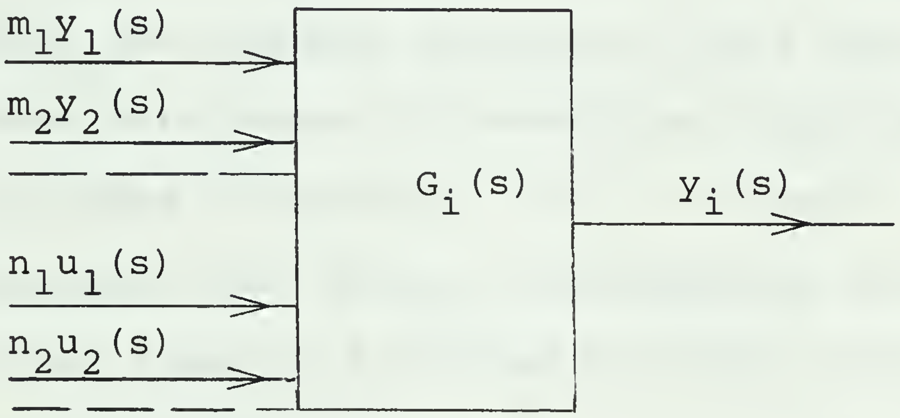


FIGURE 4-1. - General Representation of a Unit Block Diagram.

4.1 Transfer Function Definition

A general expression for $G_i(s)$ is:

$$G_i(s) = \frac{ke^{-\tau_d s}(a_m s^m + a_{m-1}s^{m-1} + \ldots . a_1 s + a_0)}{b_n s^n + b_{n-1}s^{n-1} + \ldots \ldots \ldots . b_1 s + b_0} \qquad (4\text{-}1)^{(*)}$$

where a's and b's are real constants. In equation (4-1) it is assumed that n is greater than m, a condition fulfilled by most physical control systems. Equation (4-1) may assume the following three special forms which will be

(*) A sampler plus zero-order-hold element, $(1-e^{-Ts})/s$, can be specified using equation (4-1).

referred to in the following mathematical treatment:

a. Equal order of numerator and denominator.

$$G_i(s) = \frac{a_m s^m + a_{m-1} s^{m-1} + \ldots . a_1 s + a_0}{b_n s^n + b_{n-1} s^{n-1} + \quad b_1 s + b_0} \quad n \geqq m \tag{4-2}$$

b. Pure or isolated time delay.

$$G_i(s) = ke^{-\tau_d s} \tag{4-3}$$

Only one transfer function of this type is allowed. When this occurs no other time delay such as those included in equation (4-1) is allowed. Note that isolated time delays can frequently be combined with other transfer functions by simple block diagram manipulation.

c. Isolated derivative terms.

$$G_i(s) = a_1 s + a_0 \tag{4-4}$$

Only one transfer function of this type is allowed. Further restrictions on the input will be mentioned in later sections. When an isolated derivative term is present then time delays cannot be combined into other transfer functions such as the one defined by equation (4-1). However one isolated time delay can be used.

Equations (4-1) to (4-4) represent acceptable transfer functions for the program. Consistent with the common assumption for transfer functions it is assumed that the

input and output variables have zero initial conditions, i.e., are "deviation" variables. This is done to simplify the following mathematical manipulation of the transfer functions into the equivalent state variable equations. Provision has been made in the program to handle non-zero initial conditions for problems that are entered directly as state matrix equations.

4.2 Generation of the State Equations

Equation (4-2) can be rewritten for convenience as:

$$G(s) = \frac{y(s)}{v(s)} = \frac{\beta_m s^m + \beta_{m-1} s^{m-1} + ..\beta_1 s + \beta_0}{s^n + \alpha_{n-1} s^{n-1} + ..\alpha_1 s + \alpha_0} \quad n \geq m \qquad (4-5)$$

where the coefficient α_n has been made equal to unity. The following general procedure is applied to generate the equivalent set of linear first order differential equations. Assuming the most general case of equation (4-5), i.e., where n = m, then the equivalent time domain representation can be written:

$$(p^n + \alpha_{n-1} p^{n-1} + \alpha_{n-2} p^{n-2} + ..\alpha_1 p + \alpha_0) y(t) = (\beta_n p^n + \beta_{n-1} p^{n-1} + ..+\beta_1 p + \beta_0) v(t)$$

$$v(t)\Big|_{t=0} = 0 \qquad (4-6)$$

$$y(t)\Big|_{t=0} = 0$$

where p = d/dt

Consider the following set of first order differential equations which will be shown to be equivalent to a single transfer function of the type (4-5)

$$\begin{aligned} y &= x_1 + b_0 v \\ \dot{x}_k &= x_{k+1} + b_k v \qquad k = 1,2,3,\ldots n-1 \\ &\ldots \\ \dot{x}_n &= -(a_0 x_1 + a_1 x_2 + \ldots + a_{n-1} x_n) + b_n v \end{aligned} \tag{4-7}$$

The constants a_i, b_i of equation (4-7) are expressible in terms of α's and β's of equation (4-6) as described below. Differentiating the first equation of the set (4-7) with respect to time yields:

$$py = \dot{x}_1 + b_0 \dot{v} \tag{4-8}$$

Substituting for $\dot{x}_1$ from the second equation of set (4-7) gives:

$$\begin{aligned} py &= (x_2 + b_1 v) + b_0 \dot{v} \qquad \text{or,} \\ py &= x_2 + b_1 v + b_0 pv \end{aligned} \tag{4-9}$$

Following the same procedure, the second and higher derivatives of y are:

$$\begin{aligned} p^2 y &= x_2 + b_1 pv + b_0 p^2 v = x_3 + b_2 v + b_1 pv + b_0 p^2 v \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ p^{n-1} y &= x_n + b_{n-1} v + b_{n-2} pv + \ldots + b_0 p^{n-1} v \\ p^n y &= -(a_0 x_1 + a_1 x_2 + \ldots + a_{n-1} x_n) + b_n v + b_{n-1} pv + \ldots b_0 p^n v \end{aligned} \tag{4-10}$$

Substituting for y, py, p^2y ....p^ny from equations (4-7) to (4-10) into equation (4-6)

$$
\begin{aligned}
&-(a_0x_1 + a_1x_2 + ..+ a_{n-1}x_n) + b_nv + b_{n-1}pv + .. b_0p^nv \\
&+ \alpha_{n-1}[x_n + b_{n-1}v + b_{n-2}pv + ... + b_0p^{n-1}v] + \\
&. \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \\
&+ \alpha_1[x_2 + b_1v + b_0pv] \\
&+ \alpha_0[x_1 + b_0v] = \\
&(\beta_np^n + \beta_{n-1}p^{n-1} + ... + \beta_1p + \beta_0)v
\end{aligned}
\tag{4-11}
$$

By comparing the two sides of equation (4-11) it follows that

$$\alpha_i = a_i \qquad i = 1,2,...n$$

and

$$
\begin{aligned}
\beta_0 &= \alpha_0b_0 + \alpha_1b_1 + ... + \alpha_{n-1}b_{n-1} + b_n \\
\beta_1 &= \alpha_1b_0 + \alpha_2b_1 + ... + \alpha_{n-1}b_{n-2} + b_{n-1} \\
&\; . \; . \; . \\
\beta_{n-1} &= \alpha_{n-1}b_0 + b_1 \\
\beta_n &= b_0
\end{aligned}
\tag{4-12}
$$

In expanded matrix form equation (4-12) becomes

$$
\begin{bmatrix} \beta_n \\ \beta_{n-1} \\ \beta_{n-2} \\ \cdot \\ \beta_0 \end{bmatrix}
=
\begin{bmatrix}
1 & 0 & 0 \; . \; . \; . \; . \; . & 0 \\
\alpha_{n-1} & 1 & 0 \; . \; . \; . \; . \; . & 0 \\
\alpha_{n-2} & \alpha_{n-1} & 1 \; . \; . \; . \; . \; . & 0 \\
\cdot \; \cdot \; \cdot & \cdot \; \cdot \; \cdot & \cdot \; \cdot \; \cdot \; \cdot \; \cdot & \cdot \\
\alpha_0 & \alpha_1 & \alpha_2 \qquad \alpha_{n-1} & 1
\end{bmatrix}
\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \cdot \\ b_n \end{bmatrix}
\tag{4-13}
$$

and in matrix notation

$$\underline{\beta} = \underline{\text{ALPHA}}\ \underline{b} \tag{4-14}$$

where ALPHA is the above n by n square matrix.
Since ALPHA is always nonsingular the equation may be solved solved for the vector of coefficients $\underline{b}$ as

$$\underline{b} = \underline{\text{ALPHA}}^{-1}\ \underline{\beta} \tag{4-15}$$

This equation along with equation

$$\underline{a} = \underline{\alpha} \tag{4-16}$$

completely defines the coefficients of the state equations defined by (4-7). The substitution of these coefficients into equation (4-7) provides the set of state equations equivalent to the general transfer function defined by equation (4-5). The illustrated procedure can obviously be repeated for each transfer function of the block diagram, the only difference being that the state variables are sequentially indexed from the last value up. The total number of state variables thus generated is equal to the order of the system or, in other terms, equal to the highest power of the denominator of the system transfer function or "characteristic equation."

4.3 Matrix Representation of State Equation

Substituting for the variable v of equations (4-5), (4-6) and (4-7) the more general form of input shown in

Figure 4-1 yields:

$$v(t) = m_1 y_1(t) + m_2 y_2(t) + \ldots + n_1 u_1(t) + n_2 u_2(t) + \ldots \tag{4-17}$$

where the

$y(t)$ are called "output variables" and the

$u(t)$ are called "external forcing functions."

$m_1, m_2, m_3, \ldots n_1, n_2, n_3, \ldots$ are real coefficients.

Substituting (4-17) into (4-7), separating the state variable equations from the configuration or output variable equations, dropping for simplicity of notation the independent variable, time, and subscripting the state variables so that they represent the state variables introduced by the ith transfer function to be converted

$$\dot{x}_k = x_{k+1} + b_k m_1 y_1 + b_k m_2 y_2 + \ldots + b_k n_1 u_1 + b_k n_2 u_2 + \ldots \qquad k = j, j+n-2$$

$$\dot{x}_{j+n-1} = -(a_0 x_j + a_1 x_{j+1} + \ldots + a_{n-1} x_{j+n-1}) + b_n m_1 y_1 + b_n m_2 y_2 + \ldots + b_n n_1 u_1 + b_n n_2 u_2 + \ldots \tag{4-18}$$

$$y_i = x_j + b_0 m_1 y_1 + b_0 m_2 y_2 + \ldots + b_0 n_1 u_1 + b_0 n_2 u_2 + \ldots \tag{4-19}$$

where $j = 1$ for $i = 1$

$$j = \left[\sum_{q=1}^{i-1} (\text{order of qth transfer function}) \right] + 1 \text{ for } i > 1$$

and q is just a "counting variable" for the summation. Note that the program assumes that y_i is the output variable of the block identified by the number i.

Equations (4-18) are identified with the following matrix equation:

$$\underline{X} = \underline{A1}\,\underline{X} + \underline{B1}\,\underline{Y} + \underline{C1}\,\underline{u} \qquad (4\text{-}20)$$

where

$\underline{X}$ is the vector of the state variables

$\underline{Y}$ is the vector of output variables and,

$\underline{u}$ is the vector of external forcing functions

$\underline{A1}$, $\underline{B1}$, $\underline{C1}$ are coefficient matrices.

In an analogous way equation (4-19) is identified with the following matrix equation:

$$\underline{Y} = \underline{F}\,\underline{X} + \underline{G}\,\underline{Y} + \underline{H}\,\underline{u} \qquad (4\text{-}21)$$

where

$\underline{F}$, $\underline{G}$ and $\underline{H}$ are coefficient matrices.

Block diagrams with one or more transfer functions of the type defined by equation (4-1) or with one defined by equations (4-3) and (4-4) are now discussed.

4.4 Block Diagrams Containing One or More Transfer Functions Including Time Delays

When a transfer function with a time delay is present and the degree of the numerator is less than that of the denominator the treatment so far developed applies entirely with the exception that equation (4-17) in effect becomes:

$$v(t-\tau_d) = m_1 y_1(t-\tau_d) + m_2 y_2(t-\tau_d) + \ldots n_1 u_1(t-\tau_d) + n_2 u_2(t-\tau_d) + \ldots \qquad (4\text{-}22)$$

In other words transfer functions which include time delays are handled by introducing delayed input variables and then handling the transfer function in the same manner as those defined by equation (4-2). From equation (4-12)

$$b_0 = \beta_n = 0 \tag{4-23}$$

Equations (4-18) and (4-19) become

$$\dot{x}_k = x_{k+1} + b_k m_1 y_1(t-\tau_d) + b_k m_2 y_2(t-\tau_d) + .$$

$$..+ b_k n_1 u_1(t-\tau_d) + b_k n_2 u_2(t-\tau_d) + ..$$

$$k = j \ , \ j+n-2 \tag{4-24}$$

$$\dot{x}_{j+n-1} = -(a_0 x_j + a_1 x_{j+1} + .. a_{n-1} x_{j+n-1}) +$$

$$+ b_n m_1 y_1(t-\tau_d) + ..+ b_n n_1 u_1(t-\tau_d) + ..$$

$$y_i = x_j \tag{4-25}$$

j being defined as for equations (4-18) and (4-19).

As transfer functions of the type represented by equations (4-1) and (4-2) may occur simultaneously in the block diagram a more general system is obtained by identifying equations (4-18) and (4-19) with the matrix equations:

$$\dot{\underline{X}} = \underline{A1}\,\underline{X} + \underline{B1}\,\underline{Y} + \underline{C1}\,\underline{u} + \underline{D1}\,\underline{Y}_{D1} + \underline{E1}\,\underline{u}_{D1} + \underline{D2}\,\underline{Y}_{D2} + \underline{E2}\,\underline{u}_{D2} + ... \tag{4-26-1}$$

$$\underline{Y} = \underline{F}\,\underline{X} + \underline{G}\,\underline{Y} + \underline{H}\,\underline{u} \tag{4-26-2}$$

where $\underline{D1}$, $\underline{E1}$, respectively, are coefficient matrices for

the vector of delayed output variables $\underline{Y}_{D1}$ and delayed external forcing functions $\underline{u}_{D1}$, associated with time delay τ_{d1}. The analogous coefficient matrices associated with time delay τ_{d2} are $\underline{D2}$, $\underline{E2}$ and each additional time delay would require two new coefficient matrices. For the algebraic system, equation (4-26-2) is more general than the one generated by equation (4-25).

4.5 <u>Block Diagrams Containing a Pure (Isolated) Time Delay</u>

A pure time delay implies only an algebraic relationship between input and output of the type

$$y_i(t)\Big|_{\text{output}} = v(t-\tau_d)\Big|_{\text{input}} \tag{4-27}$$

If no other time delay is allowed, the matrix state equation (4-20) holds for this case too. Referring to the general form for input shown in Figure 4-1 and defined by equation (4-17) the input-output relationship for a pure time delay becomes

$$\begin{aligned} y_i(t) &= m_1 y_1(t-\tau_d) + m_2 y_2(t-\tau_d) + \ldots \\ &\quad + n_1 u_1(t-\tau_d) + n_2 u_2(t-\tau_d) + \ldots \end{aligned} \tag{4-28}$$

so that considering equation (4-28) along with equation (4-19) the matrix system becomes

$$\dot{\underline{X}} = \underline{A1X} + \underline{B1Y} + \underline{C1u} \tag{4-29-1}$$

$$\underline{Y} = \underline{FX} + \underline{GY} + \underline{Hu} + \underline{L1Y}_D + \underline{M1u}_D \tag{4-29-2}$$

where $\underline{L1}$ and $\underline{M1}$ are the coefficient matrices for the delayed inputs given in equation (4-28).

4.6 Block Diagrams Containing an Isolated Derivative Action

A transfer function with a first order derivative term implies a relationship between input and output of the type:

$$y_i(t)\Big|_{\text{output}} = a_1\dot{v}(t)\Big|_{\text{input}} + a_0 v(t)\Big|_{\text{input}} \qquad (4\text{-}30)$$

or, more generally,

$$\begin{aligned} y_i(t) &= a_1 m_1 \dot{y}_1(t) + a_1 m_2 \dot{y}_2(t) + \ldots \\ &\quad \ldots + a_0 m_1 y_1(t) + a_1 m_2 y_2(t) + \ldots \end{aligned} \qquad (4\text{-}31)$$

Note that in this case the inputs have been limited to "output variables." This is done deliberately to avoid the necessity of calculating the derivatives of external forcing functions which would be undesirable if, for example, the external forcing functions were noisy process signals. The only justification for the above assumption was, however, not to further complicate the matrix manipulation part of the program.

As no time delay in the form of transfer function (4-1) is allowed in this case, the matrix system becomes:

$$\dot{\underline{X}} = \underline{A1}\,\underline{X} + \underline{B1}\,\underline{Y} + \underline{C1}\,\underline{u} \qquad (4\text{-}31\text{-}1)$$

$$\underline{Y} = \underline{F}\,\underline{X} + \underline{G}\,\underline{Y} + \underline{H}\,\underline{u} + \underline{Q2}\,\dot{\underline{Y}} \qquad (4\text{-}31\text{-}2)$$

where $\underline{Q2}$ is the matrix of coefficients $a_1 m_1$, $a_2 m_2$, ... defined by equation (4-31)

4.7 Block Diagrams Containg a Pure Time Delay and an Isolated Derivative Action

In this case the matrix system obviously contains the same coefficient matrices developed in previous sections where the terms were treated separately.

Thus:

$$\dot{\underline{X}} = \underline{A1}\,\underline{X} + \underline{B1}\,\underline{Y} + \underline{C1}\,\underline{u} \qquad (4\text{-}32\text{-}1)$$

$$\underline{Y} = \underline{F}\,\underline{X} + \underline{G}\,\underline{Y} + \underline{H}\,\underline{u} + \underline{Q2}\,\dot{\underline{Y}} + \underline{L1}\,\underline{Y}_D + \underline{M1}\,\underline{u}_D \qquad (4\text{-}32\text{-}2)$$

4.8 Manipulation of the System of State Equations into the Standard Form

The standard form of the state matrix equations used in most texts is:

$$\dot{\underline{X}} = \underline{A}\,\underline{X} + \underline{B}\,\underline{U} \qquad (4\text{-}33\text{-}1)$$

$$\underline{Y} = \underline{C}\,\underline{X} + \underline{D}\,\underline{V} \qquad (4\text{-}33\text{-}2)$$

Equations (4-26), (4-29), (4-31), (4-32) can be manipulated by means of matrix algebra to the form of equation (4-33). Some basic steps of this manipulation are common to all four sets of equations and will not be repeated. The procedure consists of:

- solving the algebraic matrix equation for (4-26-2) for vector $\underline{Y}$,

- substituting this result into the differential matrix equation (4-26-1),
- collecting the coefficient matrices of $\underline{X}$ and $\underline{u}$
- partitioning if necessary to conform with equation (4-33).

4.8.1 Standard Case

Consider the system defined previously by equation (4-26)

$$\dot{\underline{X}} = \underline{A1}\underline{X} + \underline{B1}\underline{Y} + \underline{C1}\underline{u} + \underline{D1}\underline{Y}_{D1} + \underline{E1}\underline{u}_{D1} + \underline{D2}\underline{Y}_{D2} + \underline{E2}\underline{u}_{D2} \tag{4-26-1}$$

$$\underline{Y} = \underline{F}\underline{X} + \underline{G}\underline{Y} + \underline{H}\underline{u} \tag{4-26-2}$$

Equation (4-26-2) can be solved for vector $\underline{Y}$ as shown

$$(\underline{I}-\underline{G})\underline{Y} = \underline{F}\underline{X} + \underline{H}\underline{u} \tag{4-34}$$

where $\underline{I}$ is the identity matrix.

The matrix $(\underline{I}-\underline{G})$ is generally* non-singular, hence the inverse exists. Premultiplying both sides of equation (4-34) by $(\underline{I}-\underline{G})^{-1}$ yields

$$\underline{Y} = (\underline{I}-\underline{G})^{-1}\underline{F}\underline{X} + (\underline{I}-\underline{G})^{-1}\underline{H}\underline{u} \qquad \text{or}$$

$$\underline{Y} = \underline{F1}\underline{X} + \underline{H1}\underline{u} \tag{4-35}$$

where
$$\underline{F1} = (\underline{I}-\underline{G})^{-1}\underline{F}$$
$$\underline{H1} = (\underline{I}-\underline{G})^{-1}\underline{H}$$

---

*Exceptions are, for example, a unity positive feedback to a block with transfer function = 1 in which case a diagonal element becomes zero.

Substituting (4-35) into (4-26-1) and collecting like matrices

$$\dot{\underline{X}} = (\underline{A1} + \underline{B1}\,\underline{F1})\underline{X} + (\underline{C1} + \underline{B1}\,\underline{H1})\underline{u} + \underline{D1}\,\underline{Y}_{D1} + \underline{E1}\,\underline{u}_{D1} + $$

$$+ \underline{D2}\,\underline{Y}_{D2} + \underline{E2}\,\underline{u}_{D2} \qquad (4\text{-}36)$$

Delaying both sides of equation (4-35) by τ_{d1} and τ_{d2}

$$\underline{Y}_{D1} = \underline{F1}\,\underline{X}_{D1} + \underline{H1}\,\underline{u}_{D1} \qquad (4\text{-}37\text{-}1)$$

$$\underline{Y}_{D2} = \underline{F1}\,\underline{X}_{D2} + \underline{H1}\,\underline{u}_{D2} \qquad (4\text{-}37\text{-}2)$$

Substituting (4-37) into (4-36) and collecting like matrices

$$\dot{\underline{X}} = (\underline{A1} + \underline{B1}\,\underline{F1})\underline{X} + (\underline{C1} + \underline{B1}\,\underline{H1})\underline{u} + \underline{D1}\,\underline{F1}\,\underline{X}_{D1} + (\underline{F1} + \underline{D1}\,\underline{H1})\underline{u}_{D1} +$$

$$+ \underline{D2}\,\underline{F1}\,\underline{X}_{D2} + (\underline{E2} + \underline{D2}\,\underline{H1})\underline{u}_{D2} \qquad (4\text{-}38)$$

Equation (4-38) is now of the form defined by equation (4-33-1)

$$\dot{\underline{X}} = \underline{A}\,\underline{X} + \underline{B}\,\underline{U} \qquad (4\text{-}33\text{-}1)$$

where $\underline{A} = \underline{A1} + \underline{B1}\,\underline{F1}$

(4-39)

$$\underline{B} = [(\underline{C1}+\underline{B1}\,\underline{H1}) \mid (\underline{E1}+\underline{D1}\,\underline{H1}) \mid \underline{D1}\,\underline{F1} \mid (\underline{E2}+\underline{D2}\,\underline{H1}) \mid \underline{D2}\,\underline{F1}]$$

$$\underline{U} = \begin{bmatrix} \underline{u} \\ \hline \underline{u}_{D1} \\ \hline \underline{X}_{D1} \\ \hline \underline{u}_{D2} \\ \hline \underline{X}_{D2} \end{bmatrix}$$

Equation (4-35) is of the form defined by equation (4-33-2)

$$\underline{Y} = \underline{C}\underline{X} + \underline{D}\underline{V} \qquad (4\text{-}33\text{-}2)$$

where

$$\underline{C} = \underline{F1}$$
$$\underline{D} = \underline{H1}$$
$$\underline{V} = \underline{u}$$

The desired state matrix equations have therefore been found in terms of the working matrices built up from the original transfer function data.

4.8.2. Isolated Time Delay

The matrix equations resulting from systems containing isolated time delays were given as

$$\dot{\underline{X}} = \underline{A1}\underline{X} + \underline{B1}\underline{Y} + \underline{C1}\underline{u} \qquad (4\text{-}29\text{-}1)$$

$$\underline{Y} = \underline{F}\underline{X} + \underline{G}\underline{Y} + \underline{H}\underline{u} + \underline{L1}\underline{Y}_D + \underline{M1}\underline{u}_D \qquad (4\text{-}29\text{-}2)$$

With respect to equation (4-29-2) the first step is analogous to the one described for the previous case.

$$(\underline{I}-\underline{G})\underline{Y} = \underline{FX} + \underline{Hu} + \underline{L1Y}_D + \underline{M1u}_D \qquad (4\text{-}41\text{-}1)$$

$$\underline{Y} = (\underline{I}-\underline{G})^{-1}\underline{FX} + + (\underline{I}-\underline{G})^{-1}\underline{Hu} + + (\underline{I}-\underline{G})^{-1}\underline{Y}_D + $$

$$+ (\underline{I}-\underline{G})^{-1}\underline{M1u}_D \qquad \text{or,}$$

in simplified notation

$$\underline{Y} = \underline{F1X} + \underline{H1u} + \underline{L2Y}_D + \underline{M2u}_D \qquad (4\text{-}41\text{-}2)$$

Because of the nature of a pure time delay, non zero rows in matrices $\underline{L2}$ and $\underline{M2}$ (or else, $\underline{L1}$ and $\underline{M1}$) correspond to zero rows in matrices $\underline{F1}$ and $\underline{H1}$ (or else, $\underline{F}$ and $\underline{H}$). More generally, the product of matrices $\underline{L2}$ and $\underline{M2}$ by the transpose matrix $\underline{F1}'$ of $\underline{F1}$ or $\underline{H1}'$ or $\underline{H}$ results in a null matrix, i.e.,

$$\underline{L2}\underline{F1}' = \underline{L2}\underline{H1}' = \underline{M2}\underline{F1}' = \ldots = \underline{0} \qquad (4\text{-}42)$$

This condition allows equation (4-41) to be separated into two parts:

$$\underline{I1Y} = \underline{F1X} + \underline{H1u} \qquad (4\text{-}43\text{-}1)$$

$$\underline{I2Y} = \underline{L2Y}_D + \underline{M2u}_D \qquad (4\text{-}43\text{-}2)$$

where $\underline{I1} + \underline{I2} = \underline{I}$ (identity matrix)

Delaying equation (4-43-1) and premultiplying the result by $\underline{L2}$ yields

$$\underline{L2Y}_D = \underline{L2F1X}_D + \underline{L2H1u}_D \qquad (4\text{-}44)$$

Adding (4-43-1) to (4-43-2) and substituting $\underline{L2}\underline{Y}_D$ with the expression given in (4-44)

$$\underline{Y} = \underline{F1}\underline{X} + \underline{H1}\underline{u} + (\underline{M2} + \underline{L2}\underline{H1})\underline{u}_D + \underline{L2}\underline{F1}\underline{X}_D \qquad (4\text{-}45)$$

Premultiplying (4-45) by $\underline{B1}$ and substitititing the result into equation (4-29-1) in place of $\underline{Y}$ yields:

$$\dot{\underline{X}} = (\underline{A1} + \underline{B1}\underline{F1})\underline{X} + (\underline{C1} + \underline{B1}\underline{H1})\underline{u} +$$

$$+ \underline{B1}(\underline{M2} + \underline{L2}\underline{H1})\underline{u}_D + \underline{B1}\underline{L2}\underline{F1}\underline{X}_D \qquad (4\text{-}46)$$

Equation (4-46) is of the desired type as expressed in equation (4-33-1)

$$\dot{\underline{X}} = \underline{A}\underline{X} + \underline{B}\underline{U} \qquad (4\text{-}33\text{-}1)$$

where $\underline{A} = \underline{A1} + \underline{B1}\underline{F1}$

$$\underline{B} = [(\underline{C1} + \underline{B1}\underline{H1}) \mid \underline{B1}(\underline{M2} + \underline{L2}\underline{H1}) \mid \underline{B1}\underline{L2}\underline{F1}] \qquad (4\text{-}47)$$

$$\underline{U} = \begin{bmatrix} \underline{u} \\ \hline \underline{u}_D \\ \hline \underline{X}_D \end{bmatrix}$$

Equation (4-45) is also of the desired type

$$\underline{Y} = \underline{C}\underline{X} + \underline{D}\underline{V} \qquad (4\text{-}33\text{-}2)$$

where

$$\underline{C} = \underline{F1} \quad ; \quad \underline{D} = [\underline{H1} \mid (\underline{M2} + \underline{L2}\underline{H1}) \mid \underline{L2}\underline{F1}]$$

$$\underline{V} = \begin{bmatrix} \underline{u} \\ \hline \underline{u}_D \\ \hline \underline{X}_D \end{bmatrix}$$

4.8.3 Isolated Derivative Action

For systems containing isolated derivative terms the appropriate matrix equations derived earlier are:

$$\dot{\underline{X}} = \underline{A1}\,\underline{X} + \underline{B1}\,\underline{Y} + \underline{C1}\,\underline{u} \qquad (4\text{-}31\text{-}1)$$

$$\underline{Y} = \underline{F}\,\underline{X} + \underline{G}\,\underline{Y} + \underline{H}\,\underline{U} + \underline{Q2}\,\dot{\underline{Y}} \qquad (4\text{-}31\text{-}2)$$

The mathematical manipulation to reduce equation (4-31) to standard form is as follows

$$(\underline{I}-\underline{G})\,\underline{Y} = \underline{F}\,\underline{X} + \underline{H}\,\underline{U} + \underline{Q2}\,\dot{\underline{Y}} \quad \text{or}$$

$$\underline{Y} = (\underline{I}-\underline{G})^{-1}\underline{F}\,\underline{X} + (\underline{I}-\underline{G})^{-1}\underline{H}\,\underline{u} + (\underline{I}-\underline{G})^{-1}\,\underline{Q2}\,\dot{\underline{Y}} \quad \text{or}$$

in simplified notation

$$\underline{Y} = \underline{F1}\,\underline{X} + \underline{H1}\,\underline{u} + \underline{Q3}\,\dot{\underline{Y}} \qquad (4\text{-}50)$$

Matrix $\underline{Q3}$ has the same properties with respect to matrices $\underline{F1}$ and $\underline{H1}$ as matrices $\underline{M2}$ and $\underline{L2}$ of the previous section, i.e.,

$$\underline{Q3}\,\underline{F1}' = \underline{Q3}\,\underline{H1}' = \underline{0}$$

In addition, to avoid the presence of derivatives of external forcing functions the following must be true

$$\underline{Q3}\,\underline{H1} = \underline{0} \qquad (4\text{-}51)$$

This equation implies that no algebraic relationship can exist between external forcing functions and input to a derivative action transfer function, and limits therefore the location of an isolated derivative action in the block diagram. Under these assumptions equation (4-50) can be differentiated to give

$$\dot{\underline{Y}} = \underline{F1}\,\dot{\underline{X}} + \underline{H1}\,\dot{\underline{u}} \qquad (4\text{-}52)$$

Premultiplying equation (4-52) by $\underline{Q3}$, substituting the result into equation (4-50)

$$\underline{Y} = \underline{F1}\,\underline{X} + \underline{H1}\,\underline{u} + \underline{Q3}\,\underline{F1}\,\dot{\underline{X}} \qquad (4\text{-}53)$$

Equation (4-53) after being premultiplied by $\underline{B1}$ can be substituted into (4-31-1).

$$\dot{\underline{X}} = (\underline{A1} + \underline{B1}\,\underline{F1})\underline{X} + (\underline{C1} + \underline{B1}\,\underline{H1})\underline{u} + \underline{B1}\,\underline{Q3}\,\underline{F1}\,\dot{\underline{X}} \quad \text{or}$$

$$(\underline{I}-\underline{B1}\,\underline{Q3}\,\underline{F1})\dot{\underline{X}} = (\underline{A1} + \underline{B1}\,\underline{F1})\underline{X} + (\underline{C1} + \underline{B1}\,\underline{H1})\underline{u} \qquad (4\text{-}54)$$

Premultiplying both sides of equation (4-54) by $(\underline{I}-\underline{B1}\,\underline{Q3}\,\underline{F1})^{-1}$

$$\dot{\underline{X}} = (\underline{I}-\underline{B1}\,\underline{Q3}\,\underline{F1})^{-1}(\underline{A1} + \underline{B1}\,\underline{F1})\underline{X} + (\underline{I}-\underline{B1}\,\underline{Q3}\,\underline{F1})^{-1}(\underline{C1} + \underline{B1}\,\underline{H1})\underline{u} \qquad (4\text{-}55)$$

Equation (4-55) is of the type

$$\dot{\underline{X}} = \underline{A}\underline{X} + \underline{B}\underline{U} \qquad (4\text{-}33\text{-}1)$$

where $$\underline{A} = (\underline{I}-\underline{B1}\underline{Q3}\underline{F1})^{-1}(\underline{A1} + \underline{B1}\underline{F1})$$

$$\underline{B} = (\underline{I}-\underline{B1}\underline{Q3}\underline{F1})^{-1}(\underline{C1} + \underline{B1}\underline{H1})$$

$$\underline{U} = \underline{u} \tag{4-56}$$

Finally substituting equation (4-33-1) into (4-53) and collecting coefficient matrices

$$\underline{Y} = (\underline{F1} + \underline{Q3}\underline{F1}\underline{A})\underline{X} + (\underline{H1} + \underline{Q3}\underline{F1}\underline{B})\underline{u} \tag{4-57}$$

This equation is of the desired type

$$\underline{Y} = \underline{C}\underline{X} + \underline{D}\underline{V} \tag{4-33-2}$$

where $$\underline{C} = \underline{F1} + \underline{Q3}\underline{F1}\underline{A}$$

$$\underline{D} = \underline{H1} + \underline{Q2}\underline{F1}\underline{B}$$

$$\underline{V} = \underline{u} \tag{4-58}$$

4.8.4. Isolated Time Delay plus Derivative Action Terms

This case will not be discussed as it merely is a combination of the last two.

4.9 Solution of the Matrix State Equation

Saucedo and Schiring (23) discuss several methods of solving the matrix state equation for continuous and discrete systems in Chapters 12 and 13, respectively, of their text. The following sections summarize the methods used in this work.

The complete solution of the matrix state equation

$$\dot{\underline{X}} = \underline{A}\underline{X} + \underline{B}\underline{U} \tag{4-33-1}$$

$$\underline{Y} = \underline{C}\underline{X} + \underline{D}\underline{V} \tag{4-33-2}$$

is given by

$$\underline{X}(t) = \underline{\phi}(t-\tau)\underline{X}(\tau) + \int_{\tau}^{t} \underline{\phi}(t-\lambda)\underline{B}\underline{U}(\lambda)\,d\lambda \quad \text{for } t \geqq \tau \tag{4-59}$$

Where $\underline{\phi}(t-\tau)$ is the fundamental or state transition matrix defined by the infinite series

$$\underline{\phi}(t-\tau) = e^{\underline{A}(t-\tau)} = \underline{I} + \frac{\underline{A}(t-\tau)}{1!} + \frac{\underline{A}^2(t-\tau)^2}{2!} + \ldots$$

$$\ldots + \frac{\underline{A}^n(t-\tau)^n}{n!} \tag{4-60}$$

and τ is the initial time.

4.10 Matrix State Difference Equation

The dependence of the state of the system at the k + 1 instant on the previous (kth) state, the previous input and the time interval (sampling period) is

$$\underline{X}_{k+1} = \underline{\phi}(T)\underline{X}_k + \underline{H}(T)\underline{U}(kT) \qquad k = 0, 1, \ldots \tag{4-61}$$

where $\underline{\phi}(T)$ is the fundamental matrix calculated at the time (sampling period) T, and

$$\underline{H}(T) = \int_{0}^{T} \underline{\phi}(t)\underline{B}dt \tag{4-62}$$

Equation (4-61) along with the corresponding discrete version of (4-33-2)

$$\underline{Y}_k = \underline{C}\underline{X}_k + \underline{D}\underline{V}(kT) \qquad k = 0,1,2, \ldots \tag{4-63}$$

can be used to calculate the time domain response of a system subject to external forcing functions, once the fundamental matrix $\underline{\phi}(T)$ has been evaluated.

4.11 Evaluation of the Fundamental Matrix

One way of determining the fundamental matrix is the power series method which consists essentially of evaluating equation (4-60) directly. The second method is based on the so-called Cayley-Hamilton Technique and is described in the following section.

4.12 The Cayley-Hamilton Technique

In this technique the matrix function, such as $e^{\underline{A}t}$ where $\underline{A}$ is a square matrix of order n, is first expressed as an equivalent matrix polynomial of degree n-1. The coefficients of this polynomial are then determined by solving a set of n linear algebraic equations which are functions of the eigenvalues of $\underline{A}$. This technique is based on the homonymous theorem which states that any square matrix $\underline{A}$ satisfies its own characteristic equation, i.e., if $P(\lambda)$ is the characteristic polynomial of $\underline{A}$, then

$$P(\underline{A}) = \underline{0} \tag{4-64}$$

Now let $N(\underline{A})$ be a matrix polynomial of higher degree than

"n", the order of matrix $\underline{A}$. Then the polynomial $N(\lambda)$ can be expressed as

$$N(\lambda) = Q(\lambda)P(\lambda) + R(\lambda) \tag{4-65}$$

where $R(\lambda)$ is the remainder polynomial of degree n-1 and $Q(\lambda)$ is the quotient polynomial.

However, since $P(\lambda) = 0$ for $\lambda = \lambda_i$ $i = 1,2, \ldots n$ it follows that under the same conditions

$$N(\lambda) = R(\lambda) \tag{4-66}$$

Correspondingly the matrix polynomial $N(\underline{A})$ which is of order greater than $P(\underline{A})$ can be written as

$$N(\underline{A}) = Q(\underline{A})P(\underline{A}) + R(\underline{A}) \tag{4-67-1}$$

and since $P(\underline{A}) = \underline{0}$ by the Cayley-Hamilton theorem, then

$$N(\underline{A}) = R(\underline{A}) \tag{4-67-2}$$

where $R(\underline{A})$ is a matrix polynomial of degree n-1 defined as the remainder polynomial after division of $N(\lambda)$ by $P(\lambda)$.

In particular the infinite series given by equation (4-60) which is rewritten with the time variable dropped for simplicity of notation, has the form:

$$e^{\underline{A}} = \underline{I} + \frac{\underline{A}}{1!} + \frac{\underline{A}^2}{2!} + \frac{\underline{A}^3}{3!} + \ldots \frac{\underline{A}^j}{j!} + \ldots \tag{4-68}$$

Being absolutely convergent, this equation can be expressed using (4-67-2), as a matrix polynomial of degree n-1:

$$N(\underline{A}) = e^{\underline{A}} = \alpha_0\underline{I} + \alpha_1\underline{A} + \alpha_2\underline{A}^2 + \ldots\alpha_{n-1}\underline{A}^{n-1} \tag{4-69}$$

and

$$N(\lambda_i) = R(\lambda_i) = e^{\lambda_i} = \alpha_0 + \alpha_1\lambda_i + \alpha_2\lambda_i^2 + \ldots$$

$$+ \ldots \alpha_{n-1}\lambda_i^{n-1} \qquad (4\text{-}69\text{-}1)$$

The real coefficients $\alpha_0, \alpha_1, \ldots \alpha_n$ of equation (4-69-1) are calculated by generating a set of n, linear algebraic equations as functions of the eigenvalues λ_i, i = 1, 2...n. For the case where all n eigenvalues of the matrix $\underline{A}$ are distinct then the following set of equations is generated directly from equation (4-66)

$$N(\lambda_1) = R(\lambda_1) = e^{\lambda_1} = \alpha_0 + \alpha_1\lambda_1 + \alpha_2\lambda_1^2 + \ldots\alpha_{n-1}\lambda_1^{n-1}$$

$$N(\lambda_2) = R(\lambda_2) = e^{\lambda_2} = \alpha_0 + \alpha_1\lambda_2 + \alpha_2\lambda_2^2 + \ldots\alpha_{n-1}\lambda_2^{n-1}$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$N(\lambda_n) = R(\lambda_n) = e^{\lambda_n} = \alpha_0 + \alpha_1\lambda_n + \alpha_2\lambda_n^2 + \ldots\alpha_{n-1}\lambda_n^{n-1} \qquad (4\text{-}70)$$

For the case where an eigenvalue λ_i is repeated s_i times (multiplicity s_i) then only one linear independent equation in λ_i can be obtained by substituting λ_i into equation (4-69-1). The remainder s_i-1 equations associated with the repeated eigenvalue λ_i are found by differentiating both sides of equation (4-66)

$$\left.\frac{d^k N(\lambda)}{d\lambda^k}\right|_{\lambda=\lambda_i} = \left.\frac{d^k R(\lambda)}{d\lambda^k}\right|_{\lambda=\lambda_i} \qquad k = 1,2,\ldots s_i-1 \qquad (4\text{-}71)$$

For the special case of an exponential matrix function, the differentiation is performed in equation (4-69-1).

Complex eigenvalues may also occur. As matrix $\underline{A}$ is a real matrix they will be complex conjugate eigenvalues.

Let $\quad \lambda_i = a + jb \quad$ and $\quad \lambda_{ic} = a - jb$

be a pair of complex conjugate eigenvalues.

From equation (4-70) follows:

$$N(\lambda_i) = e^{\lambda_i} = e^{a+jb} = e^a(\cos b + j \sin b) = \alpha_0 + \alpha_1(a + jb) + \ldots + \alpha_{n-1}(a + jb)^{n-1}$$

$$N(\lambda_{ic}) = e^{\lambda_{ic}} = e^{a-jb} = e^a(\cos b - \sin b) = \alpha_0 + \alpha_1(a - jb) + \ldots + \alpha_{n-1}(a - jb)^{n-1}$$

Separating the real and imaginary terms on the righthand side of the above equations:

$$N(\lambda_i) = e^a \cos b + j\, e^a \sin b = \alpha_0 + \alpha_1 a + \alpha_2(a^2 - b^2) + \ldots + j[0 + \alpha_1 b + 2\alpha_2 ab + \ldots]$$

$$N(\lambda_{ic}) = e^a \cos b - j\, e^a \sin b = \alpha_0 + \alpha_1 a + \alpha_2(a^2 - b^2) + \ldots - j[0 + \alpha_1 b + 2\alpha_2 ab + \ldots]$$

The imaginary terms can be eliminated defining the

following two equations:

$$N^*(\lambda_i) = \frac{N(\lambda_i) + N(\lambda_{ic})}{2} = e^a \cos b = \alpha_0 + \alpha_1 a + \\ + \alpha_2(a^2 - b^2) + \ldots \tag{4-72-1}$$

$$N^*(\lambda_{ic}) = \frac{N(\lambda_i) - N(\lambda_{ic})}{2} = e^a \sin b = 0 + \alpha_1 b + 2\alpha_2 ab + \ldots \tag{4-72-2}$$

Equations (4-72) represent two algebraic equations, in terms of $\underline{\alpha}$, associated with eigenvalues from matrix $\underline{A}$. Thus the coefficients $\underline{\alpha}$ in equation (4-69) can be determined by solving the set of n algebraic equations defined by (4-70), (4-71) and (4-72). This set of equations can be shown in matrix notation as

$$N(\lambda) = \underline{CC}\underline{\alpha} \tag{4-73}$$

where $\underline{CC}$ is an n by n square matrix which is never singular because of the peculiar way it is built.

Equation (4-73) may therefore be solved for $\underline{\alpha}$

$$\underline{\alpha} = \underline{CC}^{-1} N(\underline{\lambda}) \tag{4-74}$$

Therefore for any given matrix $\underline{A}$, the substitution of the values of $\underline{\alpha}$ calculated from equation (4-74) into equation (4-69) definesthe exponential matrix function and by analogy to equation (4-60) the fundamental or transition

matrix as

$$\underline{\phi}(t-\tau) = \sum_{j=0}^{n-1} \alpha_j \underline{A}^j (t-\tau)^j \qquad (4-75)$$

In their comparison of methods for determining the transition matrix $\underline{\phi}(T)$ Saucedo and Schiring (23, p.609) conclude that "For computer solution the power series or the Cayley-Hamilton method would probably be the first choice." Both of these techniques have been incorporated into the computer program produced as part of this thesis and a comparison of the results obtained by applying both techniques to the same problem are presented in Chapter VI, examples (6-1), (6-4) and (6-7).

The other mathematical and numerical techniques, such as numerical integration, used in the balance of the computer program are covered in basic texts and are assumed to be familiar to the reader.

Chapter V

COMPUTER IMPLEMENTATION

The program is written in FORTRAN IV language and is designed for implementation on an IBM 1800 DACS computer, as a non-process program. It utilizes the "variable" or non-process area of core storage (9.7K words, 16 bit/word), auxiliary disk storage and the following peripheral devices:

1442 card reader, punch

1443 or 1132 printer

1816 keyboard/printer

1627 plotter and/or scope

Because of its total length it was necessary to divide the program into 17 parts or "links" each of which could be executed in the available core storage. A flow diagram based on such links is given in Figure D-1, Appendix D. Systemwise each link represents a new job. Therefore the transmittal and storage of data from link to link was of basic importance and was achieved by the use of COMMON and disk FILES.

The COMMON area was defined as several blocks and variables were assigned to specific locations by equivalence statements. The structure of COMMON was thus kept relatively simple. Table D-1 gives the dimension of COMMON throughout the program. Two basic concepts were followed

in building the COMMON storage area.

a. Keep it as small as possible in each link.

b. Data utilized last were stored at the beginning of the COMMON area so they would not be affected by the frequent changes in storage assignments or area length.

These rules allowed a more flexible use of COMMON and consequently a more rational utilization of variable core.

FILES were extensively used for coefficient matrices when building the state matrix equations described in Chapter IV, Theory and Mathematical Basis. Generally each matrix was identified by a separate FILE. For FILE definition and use see Tables D-2 and D-3 of Appendix D. The matrix manipulation required to yield the state equation was carried out in three blocks in COMMON so that a minimum of space was required at any time. Clearly this restriction on the COMMON allocated for this purpose produced a great number of core-disk transfers with the obvious result of slowing down the program. In contrast the FILES of time domain response data were built piecewise by transferring a working vector filled up with data to the FILE. This was done to reduce the number of disk transfer operations, hence speeding up the process.

Although the program is presently dimensioned for a maximum of 10 state variables, 15 output variables and 10 forcing functions it can be easily scaled up by modifying the dimension, COMMON and equivalence statements plus the

size of FILES. The maximum dimensions of vectors and matrices, which are dealt with columnwise in one dimension, are generally equated to an integer/variable at the beginning of the link so that only this variable definition must be changed. All subroutines were written to be independent of variable dimensions similar to those from the IBM Scientific Subroutine Package (13) and thus need not be changed when the program is scaled up.

The program, in general, uses standard matrix manipulations and numerical methods. A few special programming features are described in the next section.

5.1 Special Programming Features

Although a classical and well-known problem the calculation of real distinct, real repeated, complex conjugate eigenvalues originating from the real, generally unsymmetric matrix $\underline{A}$ required some consideration.

The problem was solved by first finding the coefficients of the characteristic equation and then its roots by using the Bairstow method. As this method finds a second order polynomial divisor of the characteristic polynomial, it was believed that the problem of poor convergence around a multiple root and the problem of finding complex roots could be overcome. The program actually includes two subroutines that represent different implementations of the Bairstow method. They are mentioned in LOCAL (15) control cards as only one is needed

at the time.

Other subroutines for the eigenvalue problem can be easily built into the program as they become available by utilizing unused values of "flags" already included in the program. One additional feature concerning the eigenvalues is that they are checked for their sign and shifted so that the smallest eigenvalue becomes zero. Positive eigenvalues imply instability as some of the state variables become unbounded by large values of time. They are, therefore, singled out and the program may stop or continue according to an option specified by the user. The shifting of the eigenvalues was done to improve the "scaling" and reduce numerical errors. In case that the eigenvalue subroutines indicate poor convergence, the problem may still be run by using the power series method included in the program package.

The calculation of the coefficient matrix

$$\underline{H}(T) = \int_0^T \underline{\phi}(t)\underline{B}dt$$

uses a trapezoidal integration rule. The "standard" solution adopts 10 integration subintervals. Calculations for example 6-1 showed no significant improvement by increasing the number of subintervals to 100. Also a comparison with Simpson's integration over 150 subintervals did not show any significant difference.

Values of the delayed variables, such as $X_i(kT-\tau_d)$,

required in the generation of the time domain solution using equations (4-61) and (4-63) are obtained from a storage vector. For example, values of $X_i(kT)$ are stored in the vector as they are generated so that $X_i(kT-\tau_d)$can be obtained from earlier entries in the same vector. The vector is presently dimensioned as 200 and is used as a ring buffer or a circular file so there is no limit on, k, the number of solution points generated.

The details of the computer implementation can be determined by reference to the detailed flow diagram in Figure B-2 and the comment cards and program listings included in Appendices E and F. Some of the provisions for future extensions and linkages for alternative techniques are also described in Chapter VII.

Chapter VI

PROGRAM DEMONSTRATION AND APPLICATIONS

Most of the work presented so far deals with the mathematical and computational aspects of the thesis. Conversely, this chapter is meant to emphasize the program versatility in dealing with typical process control applications and its usefulness in solving real control problems.

For this purpose it may be pointed out that some graduate students of this department are already using the program for their research projects. One application is the generation of the state difference equations of a matrix system model of a double-effect evaporator installed in the department (see also Example 6-1).

A second use of the program is in conjunction with a separate optimal control program. For a system described by a state difference equation model of the type produced by this program the optimal control program generates a series of discrete values of $\underline{U}(t)$ which will drive the system from state A to state B. These discrete values plus the system equations are then entered into this program to obtain the time domain response.

The applications and use of the program are best

illustrated by examples. The examples were selected:

a. to demonstrate the utilization of the program for typical control oriented applications,

b. to demonstrate special features of the program, as for instance, its capability of handling multifeed-back-loop block diagrams, isolated time delays, isolated derivative action, etc.,

c. to demonstrate the validity of the program by comparing the results found using different methods of calculation and to illustrate the effect of some of the program parameters such as integration step sizes.

The discussion of each example is organized into the same format for ease of reference. The computer produced documentation for each example is included in Appendix C.

6.1 Example 6-1 Five Equation Evaporator Model

Control Purpose: To find the coefficient matrices of the state difference equation for use in real-time computer control and the open-loop response of a fifth order state equation model of a double effect evaporator. The derivation of the model shown in Figure 6-1 is outlined in a separate report(*).

Program Purpose:

- to check the performance of the program for a realistic matrix model.
- to check the accuracy of the solution.
- to illustrate that the steady state values for the forcing functions can be entered as step functions.

Results: The response of the outlet concentration and the holdup in the two effects caused by an inconsistent set of initial conditions is included in the listing on page C-4.

Comments: Newell(*) used Simpson's integration rule with 150 steps for a sampling interval of 1 sec. The results from this example were in good agreement with his for the same sampling interval and integration intervals of 0.1 sec.

(*) "Linear Evaporator Model", internal department research report, R. B. Newell, D. G. Fisher.

$$\begin{bmatrix} \dot{w}_1 \\ \dot{c}_1 \\ \dot{h}_1 \\ \dot{w}_2 \\ \dot{c}_2 \end{bmatrix} = \begin{bmatrix} 0 & -0.002 & -0.087 & 0 & 0 \\ 0 & -0.043 & +0.087 & 0 & 0 \\ 0 & -0.014 & -0.571 & 0 & 0 \\ 0 & -0.003 & -0.125 & 0 & -0.0001 \\ 0 & +0.039 & +0.125 & 0 & -0.036 \end{bmatrix} * \begin{bmatrix} w_1 \\ c_1 \\ h_1 \\ w_2 \\ c_2 \end{bmatrix} +$$

$$+ \begin{bmatrix} 0 & -0.045 & 0 & +0.065 & 0 & 0 \\ 0 & 0 & 0 & -0.02 & +0.045 & 0 \\ +0.169 & 0 & 0 & -0.04 & 0 & 0.025 \\ 0 & +0.061 & -0.036 & 0 & 0 & 0 \\ 0 & -0.024 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} S_i \\ B_1 \\ B_2 \\ F \\ C_F \\ h_F \end{bmatrix}$$

$$\begin{bmatrix} w_1 \\ w_2 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} w_1 \\ c_1 \\ h_1 \\ w_2 \\ c_2 \end{bmatrix}$$

FIGURE 6-1 Five Equation Evaporator Model

6.2 Example 6-2 Controller Constants from Library

Control Purpose

To determine the time domain response of the system illustrated in Figure 6-2-1 for step disturbances in set point and load. Ten different sets of "optimum" controller constants, as listed in Table 6-1, were used and the response of the output compared.

Program Purpose

- to check the standard "block diagram input" section.
- to illustrate handling of a time delay of the same order of magnitude as the process time lag.
- to show the ability of the program to supply "optimum" controller constants for specified standard processes and control criteria.
- to demonstrate the ability to rerun a problem several times by entering only the new controller constants.
- to illustrate the special option of producing multiple plots on one graph.

Results

The time domain response of the system output to a step change in set point, for eight different sets of controller constants is given in Figure 6-2-2. The response to a unit step change in load is given in Figure 6-2-3.

Comments Note the significant differences in response for different criteria. Since the error is always fractional the ISE criterion gives larger deviations than the IAE criterion.

Pages C-9 and C-10 of Appendix C are copies of the input/output messages to plot response 1 in a non-standard format and change from case 1 to 2.

Numerical values for the process transfer function parameters were within the range specified in the original papers.

Specifically the ranges of process parameter values for which this option may be used are as follows:

Criteria 1 through 6 of Table 6-1

$0 \leqslant \tau_d/\tau \leqslant 1.0.$ [reference (20)]

Criteria 7 and 8 of the same Table

$0.1 \leqslant \tau_d/\tau \leqslant 1.0$ [reference (5)]

where τ_d = delay

τ = lag

TABLE 6-1

Library Supplied Controller Constants for the PI Controller of Example 6-2

Criteria	Code	Servo Problem		Regulatory Problem	
		K_c	τ_I	K_c	τ_I
Ziegler-Nichols	1	0.1125	1.3332	--	--
3-C	2	0.1146	0.4073	--	--
IAE	3	0.1226	0.7020	--	--
ISE	4	0.1616	0.8619	--	--
ITAE	5	0.1068	0.6375	--	--
Cohen-Coon	6	0.1332	1.0347	--	--
Quickest response + 20% overshoot	7	0.0750	0.5000	0.0875	0.9333
Quickest response without overshoot	8	0.0437	0.5833	0.0750	1.6000

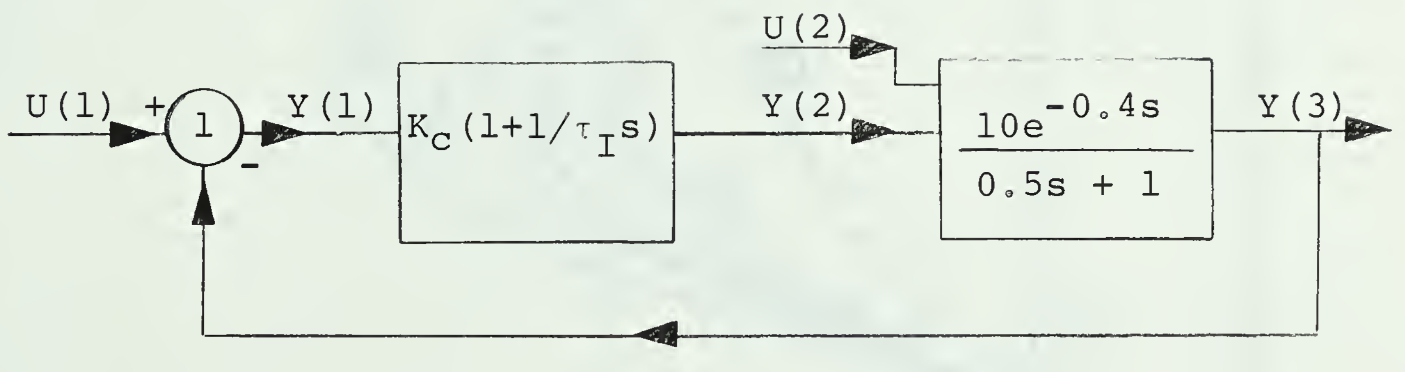


FIGURE 6-2-1 Block Diagram for Example 6-2

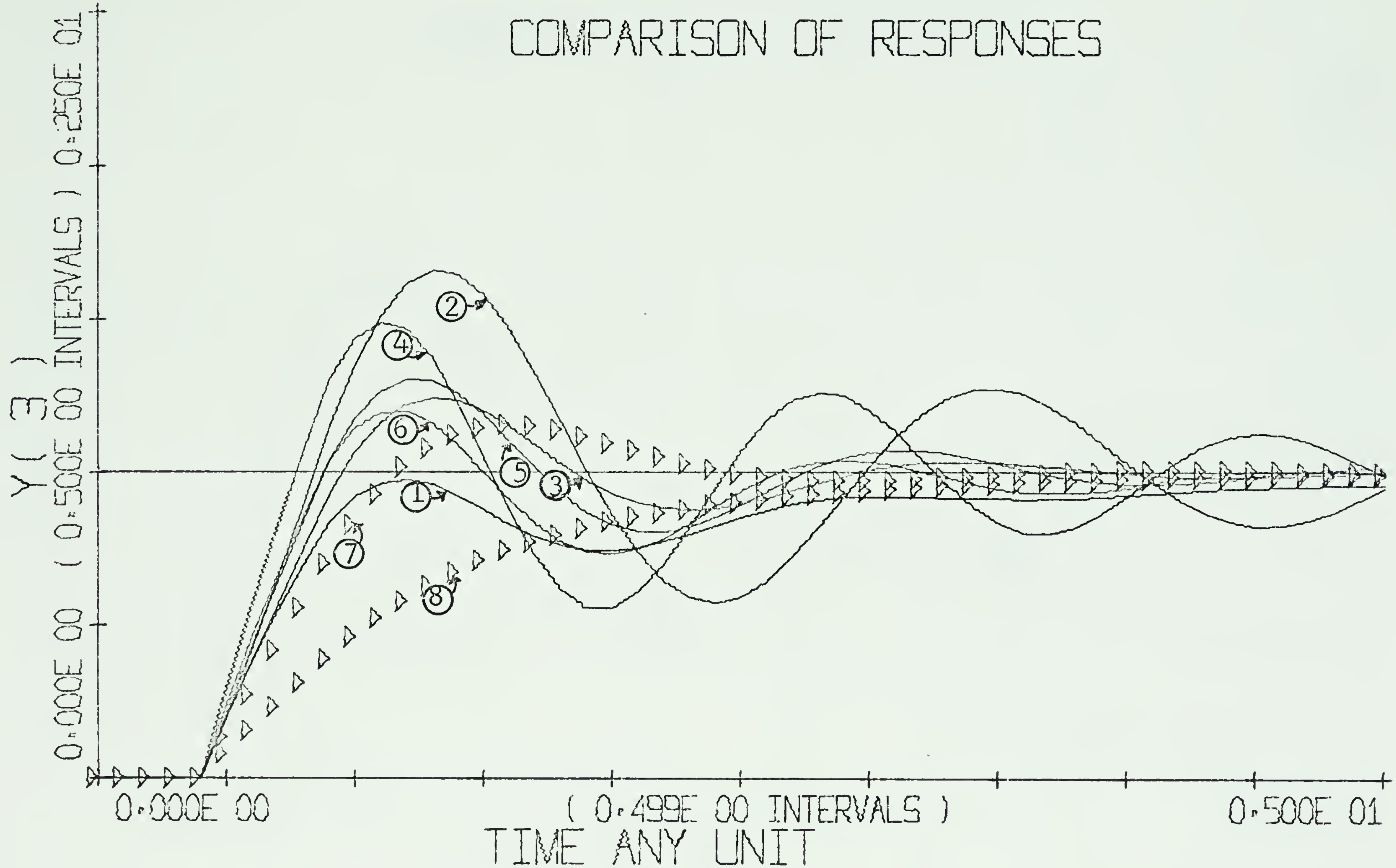


FIGURE 6-2-2 Time Domain Response for Example 6-2: Set Point Change

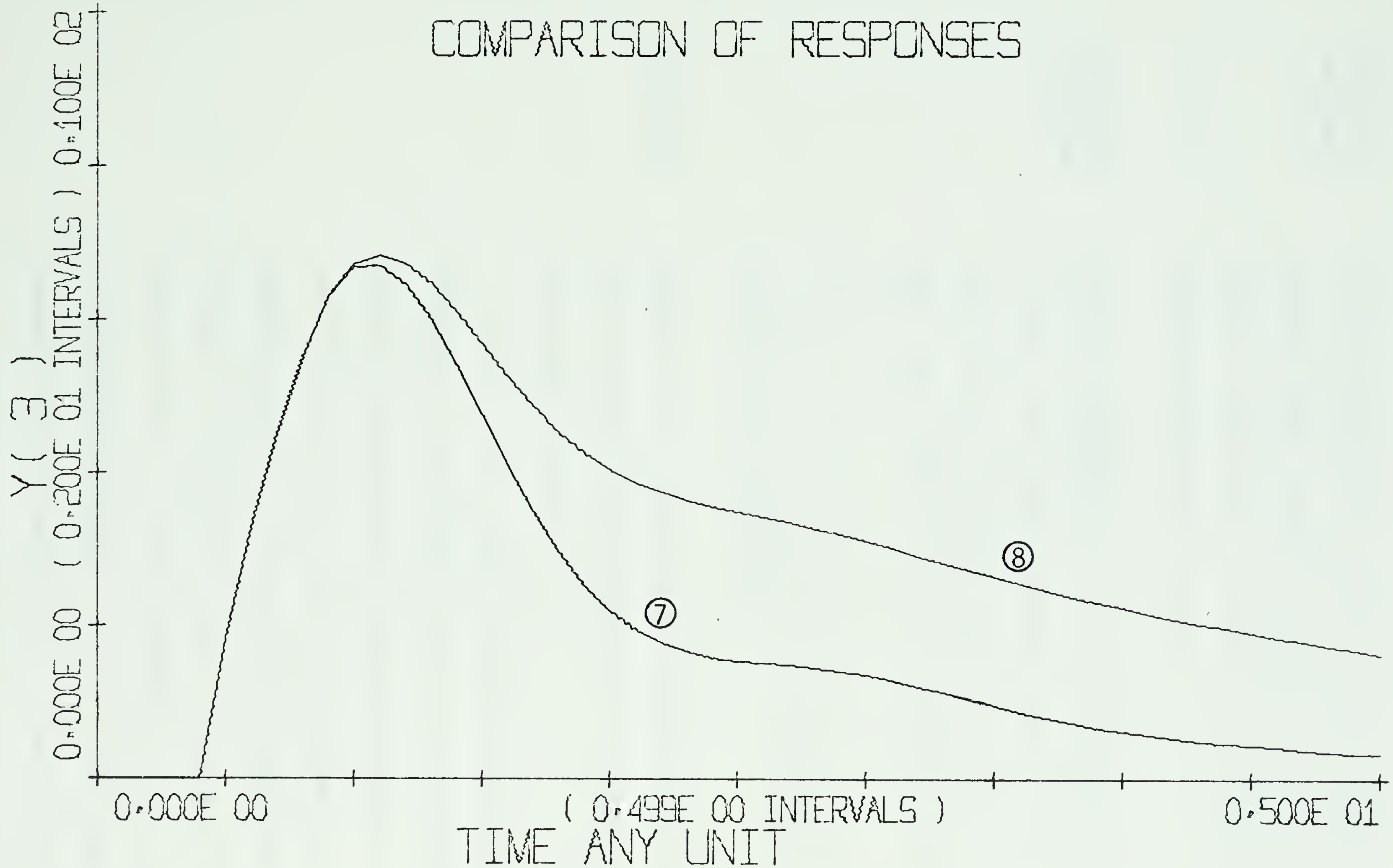


FIGURE 6-2-3 Time Domain Response for Example 6-2: Load Change

6.3 Example 6-3 Closed-Loop Tuning

Control Purpose

To illustrate the Ziegler-Nichols method for closed-loop, on-line tuning of a standard control loop such as that defined by the block diagram in Figure 6-3-1.

Program Purpose

- to test handling of an isolated time delay.
- To show that the summing junction which is normally put in the block diagram to generate the error signal can be combined with the controller.
- To test the ability of the program to handle oscillatory responses.

Results

Figure 6-3-2 shows the closed-loop responses. The ultimate gain and frequency were found from the graph to be:

$$K_u = 6.9 \; ; \; P_u = 4 \text{ time units/cycle}$$

Controller constants calculated from these values according to Ziegler-Nichols criteria are shown in Table 6-2. Responses of the same system for a P, PI and PID controller using settings of Table 6-2 are shown in Figure 6-3-3. Results are in good agreement with those found in Coughanowr-Koppel (6) for the same example (example 19.3, page 242). The output variable Y(3) in Figure 6-3-3 for the case of a

proportional controller is delayed of 0.5 time units with respect to Y(2) for the same type of controller. This confirms the proper handling by the program of isolated time delays.

Comments The ability to change the controller transfer function from an algebraic (proportional) to one which adds one or two additional state variables (PI or PID control, respectively) and to rerun the problem was again utilized in this example. Closed-loop tuning is a well known technique to estimate controller constants. It basically consists in substituting for the actual controller, a proportional one having an arbitrary (small) value for the gain. Observing the closed-loop response of the system to an external disturbance, the gain is increased until sustained oscillations are produced. This value of the "ultimate" gain and the frequency of the oscillation are used in determining the constants for a P, PI or PID controller using Ziegler-Nichols' criteria.

TABLE 6-2

Ziegler-Nichols Controller Settings for Example 6-3

Type of Controller	Controller Constants		
	K_p	τ_I	τ_D
P	3.45	--	--
PI	3.10	3.22	--
PID	4.15	2.00	0.5

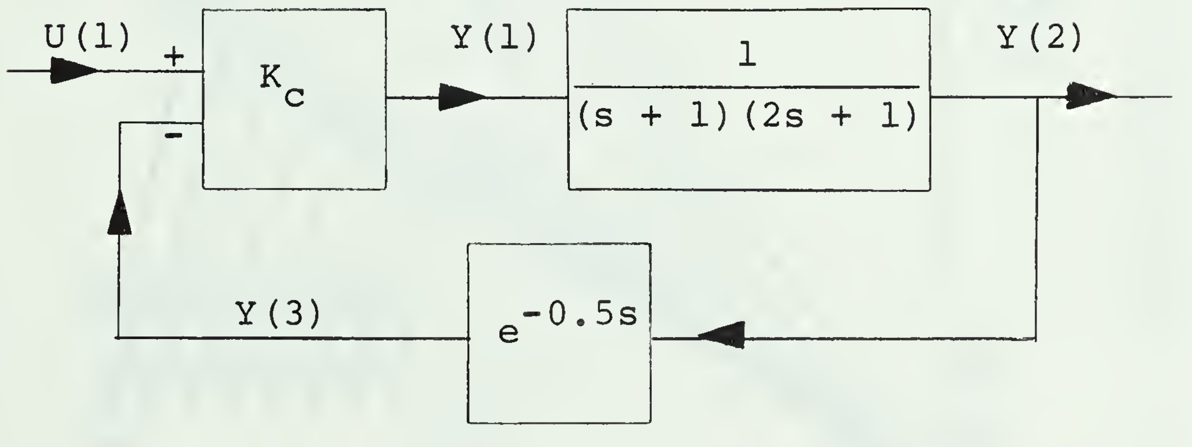


FIGURE 6-3-1 Block Diagram for Example 6-3

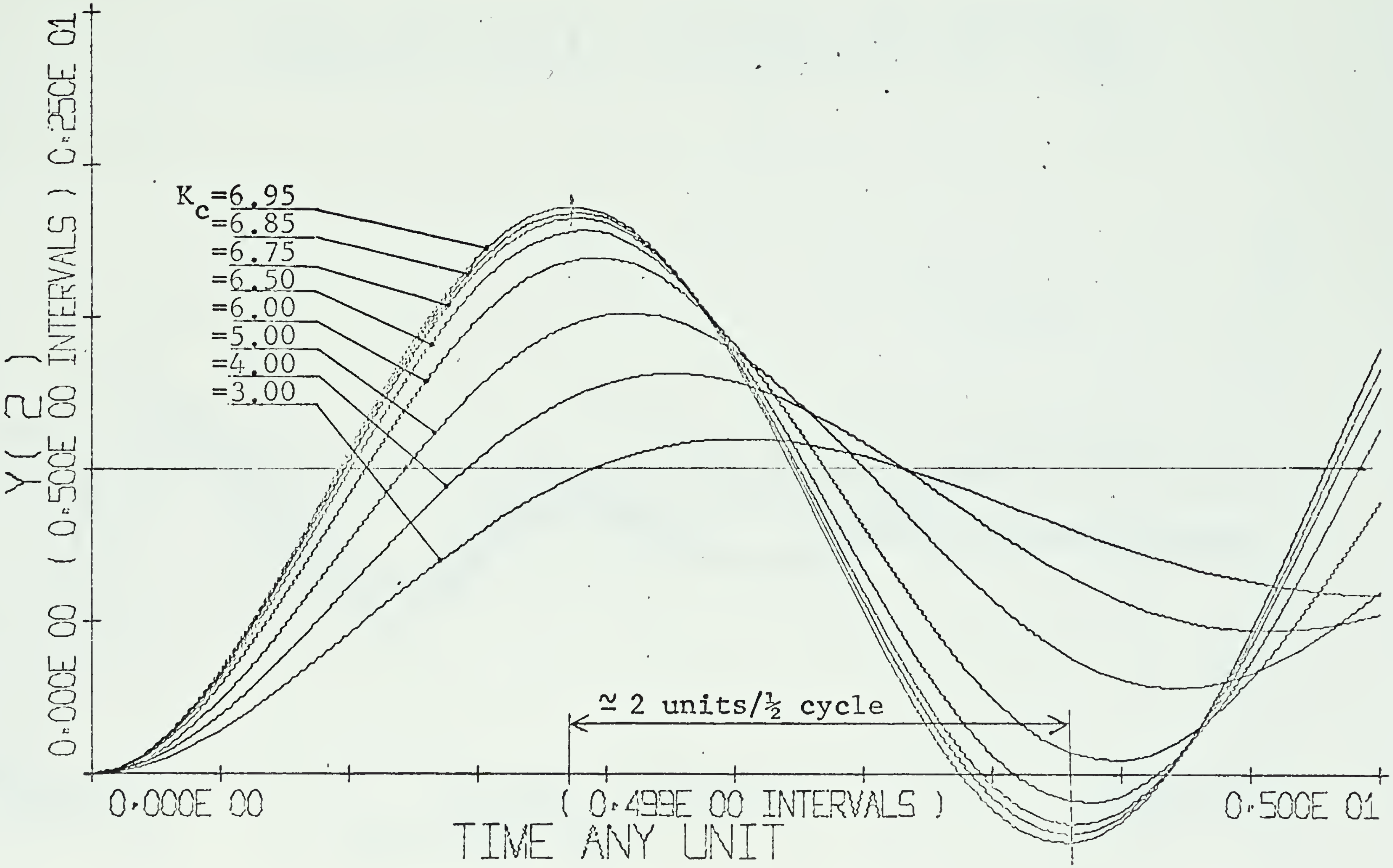


FIGURE 6-3-2 Time Domain Response for Example 6-3

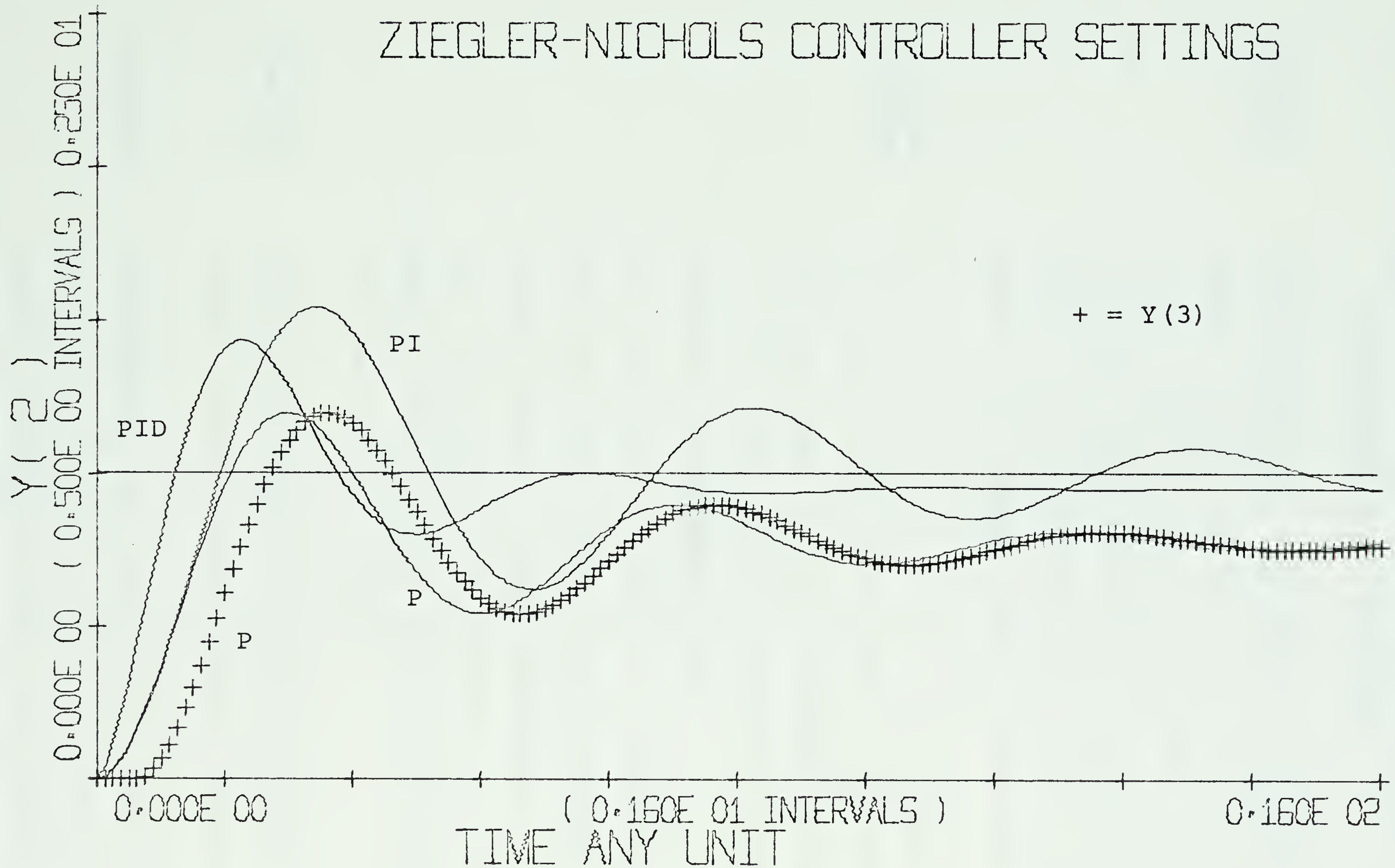


FIGURE 6-3-3 Time Domain Response for Example 6-3: Ziegler-Nichols Controller Settings

6.4 Example 6-4 Power Series Method versus Cayley-Hamilton Technique

Control Purpose

To determine the time domain response of the system defined by the block diagram in Figure 6-4-1. This system has non-unity feedback and a process transfer function containing a time delay in the numerator and a second order polynomial with complex roots in the denominator.

Program Purpose

- to confirm the validity of the Cayley-Hamilton technique in case of complex eigenvalues. This technique was, in fact, extended in this work to account for complex eigenvalues as well as distinct and repeated ones.
- to illustrate the option in the program to rerun the same problem using different calculational methods and/or parameters. In this case the method of calculating the fundamental matrix was changed from the Cayley-Hamilton to the power series option.

Results

Results are shown in Figure 6-4-2 for two values of proportional controller constant.

Comments

In comparing the results it is important to emphasize the fact that the two methods are totally unrelated so that the good agreement supports the accuracy of each method. Good

agreement between the two methods was also found in example 6-1. Differences start showing up, however, for an improper choice of the time interval because of "round-off" errors. A more detailed investigation in this respect is carried out in example 6-7. Note that the arbitrary choice of the proportional constant did not give "good" control. No attempt was made to find optimum values.

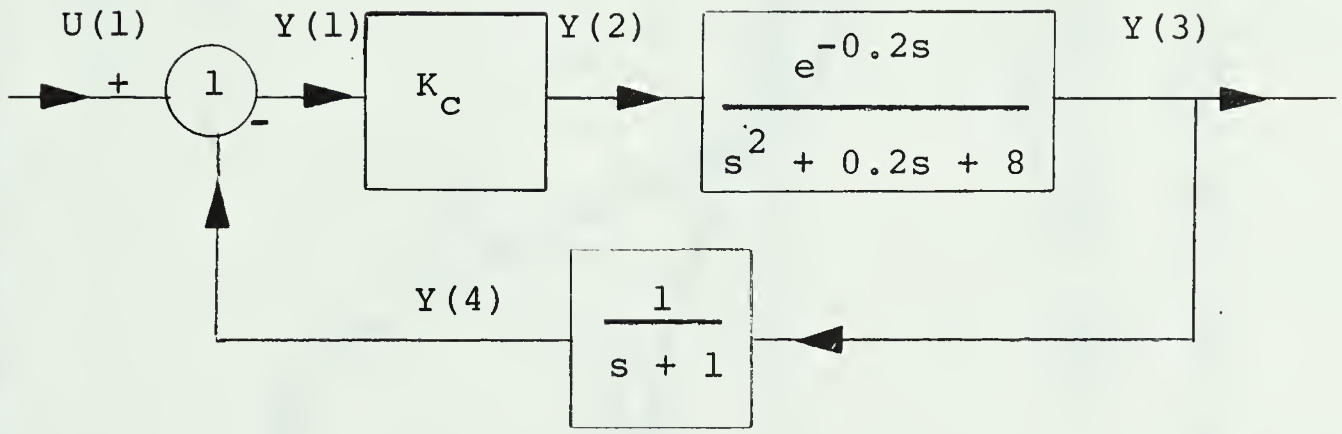


FIGURE 6-4-1 Block Diagram for Example 6-4

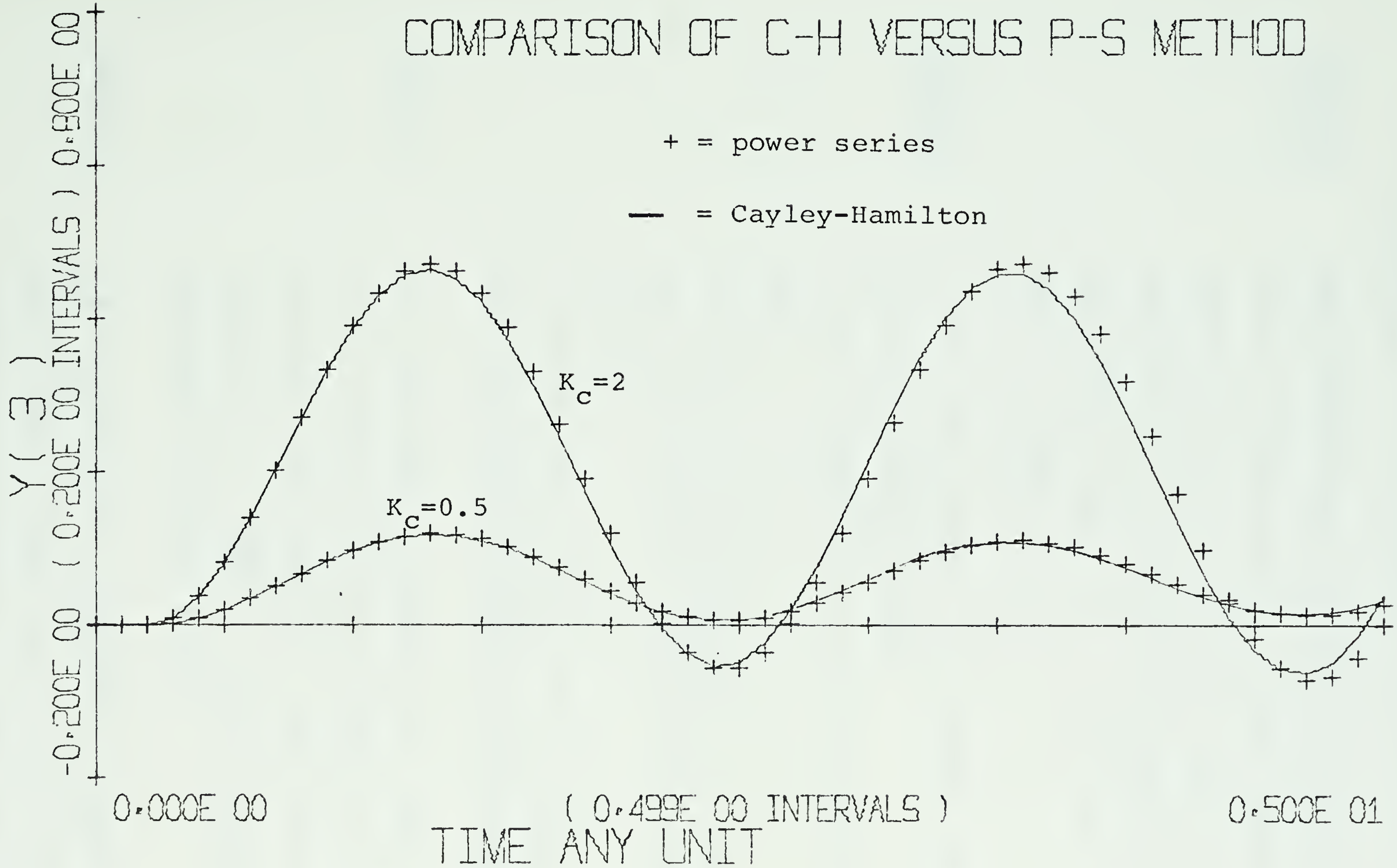


FIGURE 6-4-2 Time Domain Response for Example 6-4

6.5 Example 6-5 Multifeedback Loop Block Diagram

Control Purpose

This example was selected to show the capability of the program to handle multiloop systems as well as special types of transfer functions. It is given as exercise A-4-1, page 223 in Ogata (22). The block diagram is shown in Figure 6-5-1.

Program Purpose

- to test the program with systems that have multiple feedback loops.
- to test handling of an isolated derivative term.
- to illustrate the inclusion of a pure integral term (zero pole) in the process transfer function.
- to demonstrate accuracy of the eigenvalue and the time domain response calculations.

Results

The time domain response for a unit step change in set point is given in Figure 6-5-2 for two variables Y(4) and Y(5). From the block diagram $Y(5) = \frac{dY(4)}{dt}$: this is graphically confirmed in Figure 6-5-2.

Comments

The documentation included in Appendix C for this example is typical of a "second level" of documentation (NPRTY = 5, see User's Manual, Appendix A). This was done to exemplify this

feature of the program and to show the eigenvalues of coefficient matrix $\underline{A}$: they are the same as those given in the cited textbook.

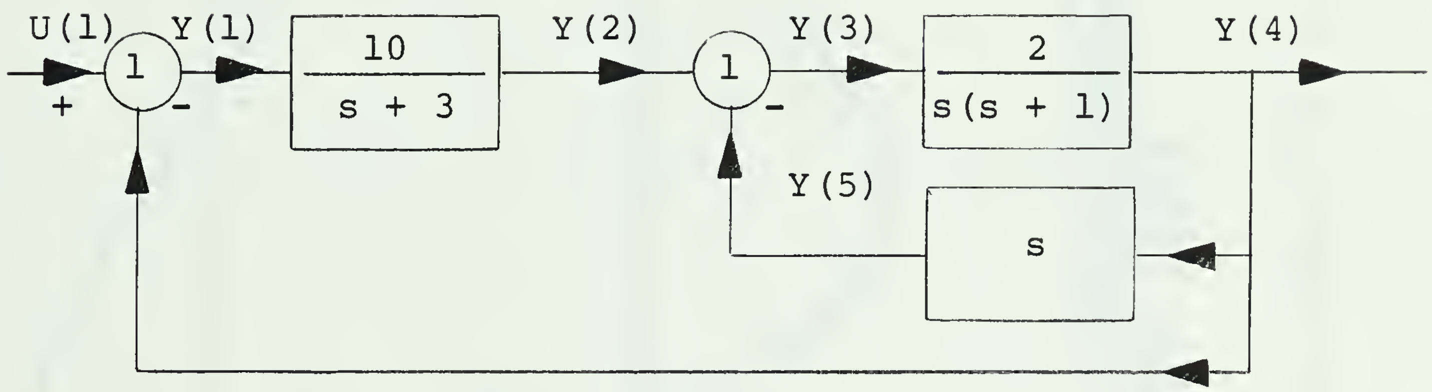


FIGURE 6-5-1 Block Diagram for Example 6-5

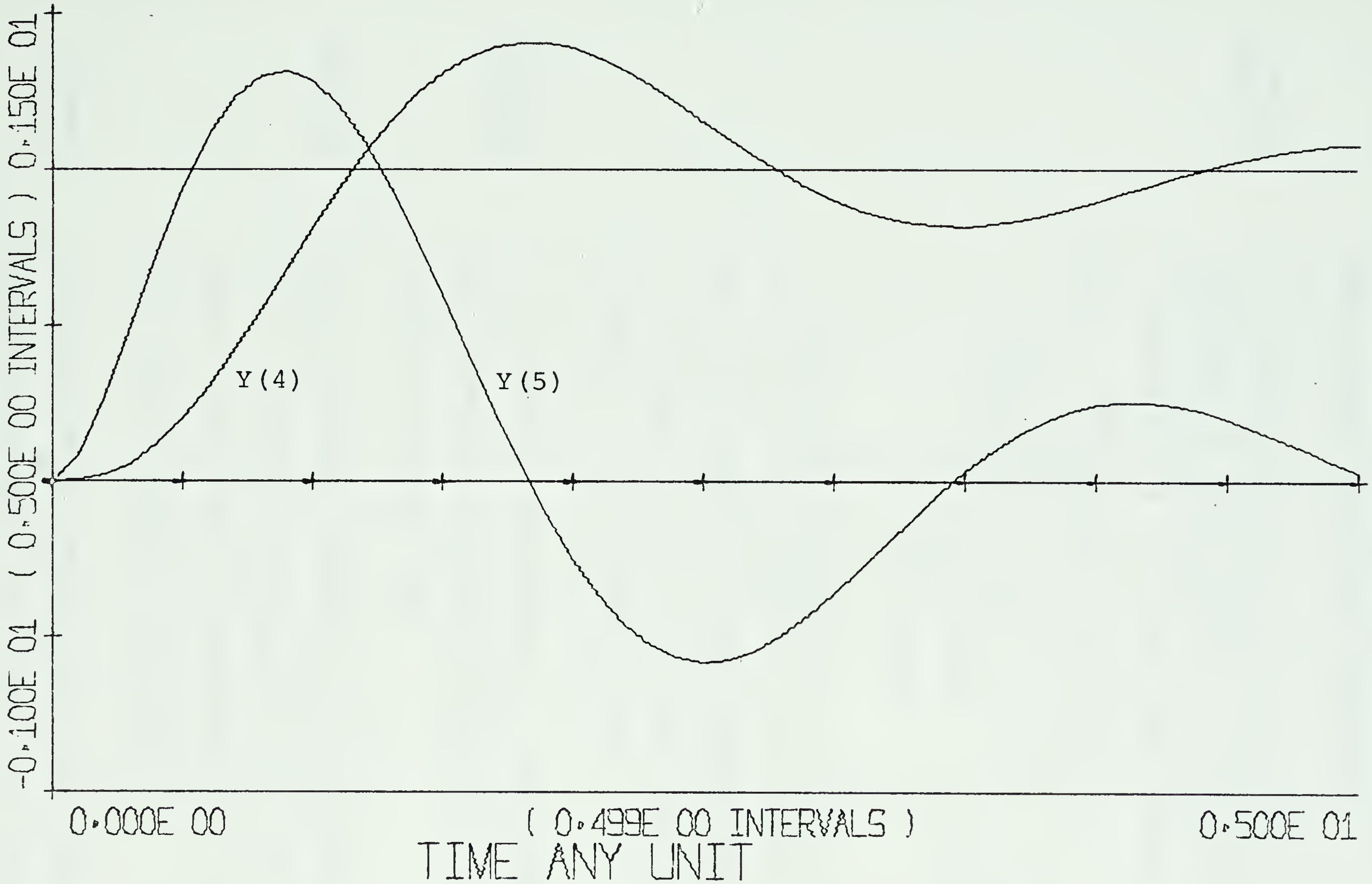


FIGURE 6-5-2 Time Domain Response for Example 6-5

6.6 Example 6-6 Effect of Negligible Time Lags on the Time Domain Response

Control Purpose

To investigate the effect of several time constants on the response of the system defined in Figure 6-6-1 to a unit step change in set points. Three cases were evaluated:

Case	G_v	G_m
1	1/(.01s+1)	1/(.001s+1)
2	1/(.01s+1)	1
3	1	1

Since the time constants of G_v and G_m were small with respect to the process time constants, it was expected that the output response would be similar for all three cases.

Program Purpose

To illustrate the ability of the program to handle systems where the eigenvalues differ by a factor of about 1000.

Results

The time domain response of the output variable Y(3) is plotted and compared for these three cases, in Figure 6-6-2 and again in Figure 6-6-3 over a larger time range.

Comments

The time domain response for each case was identical as shown by Figure 6-6-3 where all the curves are superimposed on one another.

The modification of the block diagram from Case 1 to Case 3 was made at the console keyboard by utilizing once more the user's intervention capability of changing transfer function and rerunning the program.

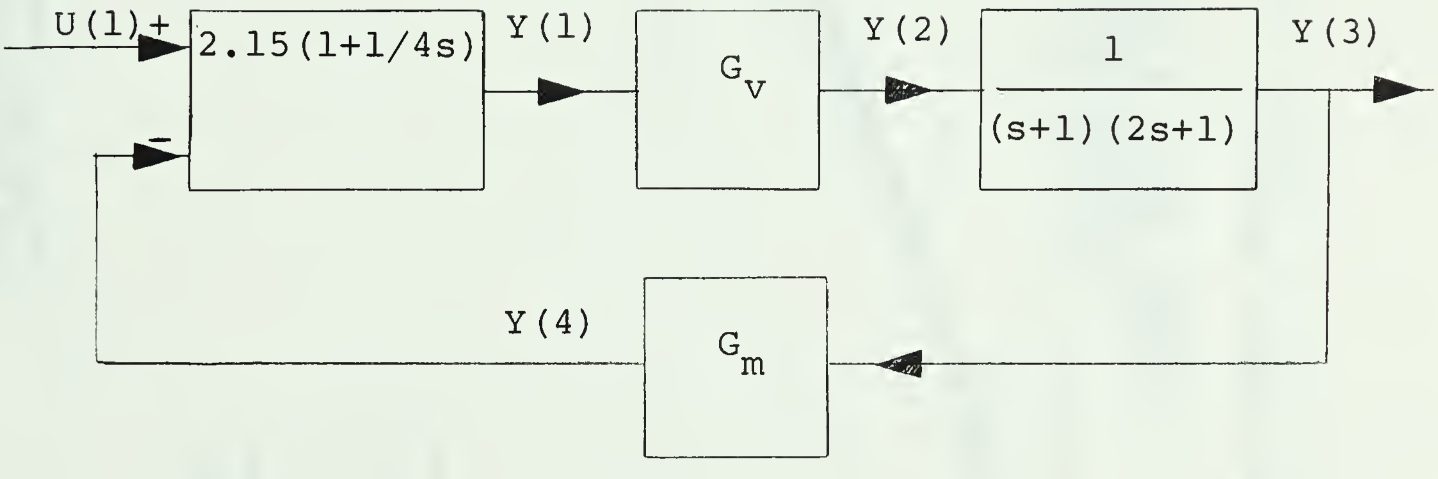


FIGURE 6-6-1 Block Diagram for Example 6-6

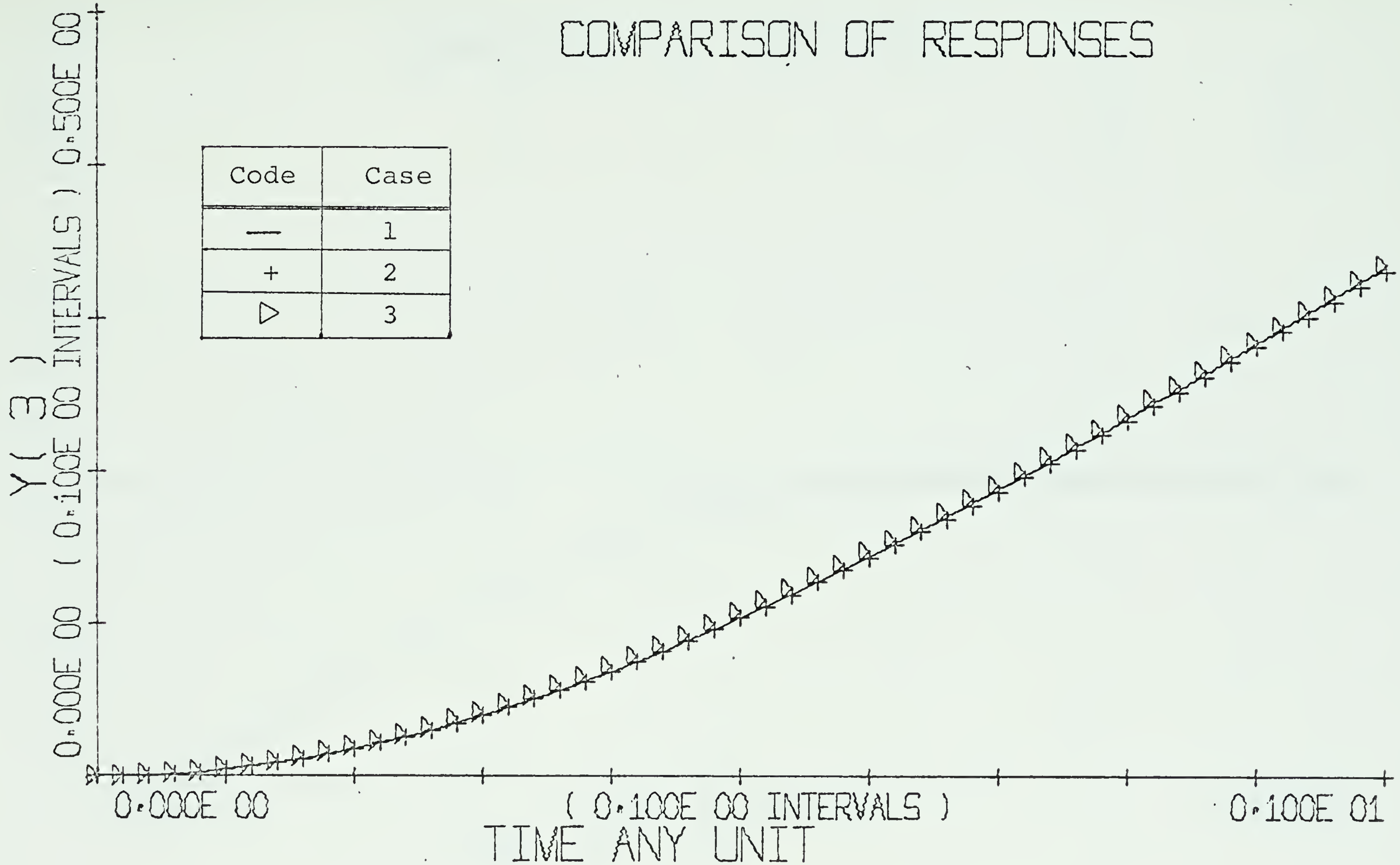


FIGURE 6-6-2 Time Domain Response for Example 6-6: Initial Response

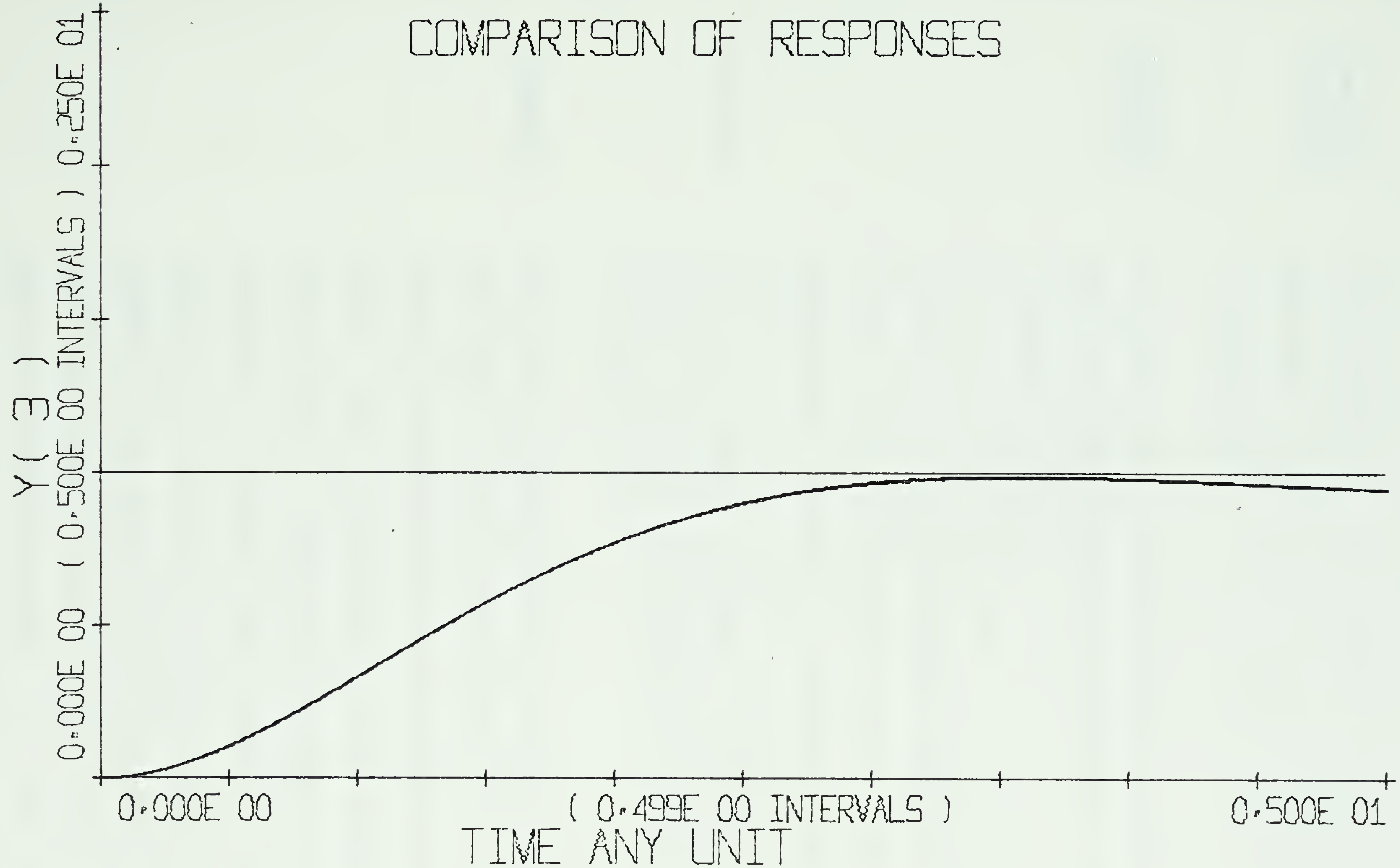


FIGURE 6-6-3 Time Domain Response for Example 6-6: Complete Response

6-7 Example 6-7 Accuracy of Calculations

Control Purpose

To investigate the response of the system defined in Figure 6-6-1 to a unit step change in set point.

Program Purpose

- to compare the Cayley-Hamilton versus the power series method for calculating the fundamental matrix.
- to investigate the effect of the integration interval on the time domain response.
- to gain experience with a system where the eigenvalues differ by a factor of 1000.

Results

The time domain response of the controlled output variable is plotted in Figures 6-7-1, 6-7-2 and 6-7-3.

Comments

The block diagram shown in Figure 6-6-1 was used again in this example because of the wide range of magnitudes of the eigenvalues (1 to 1000). This is expected as the eigenvalues are related to the time constants of the block diagram. A study of the effect of the time interval on the final response for a unit step change in set point was carried out by rerunning the program several times after having made the desired changes. In other runs the power series method was used in place of the

standard Cayley-Hamilton technique. Figure 6-7-1 shows the effect of several integration intervals on the initial response. Notice that the power series method gives identical response to the Cayley-Hamilton technique only for an integration interval of .0001 (this value is approximately equal to 1/10 of the reciprocal of the largest entry in matrix $\underline{A}$). Figure 6-7-3 extends those responses to larger time range for the Cayley-Hamilton technique only. It is interesting to point out that increasing the integration interval causes the system to behave like a physically "unstable system" whereas limited accuracy and round off errors are the true cause of the observed "instability". Figure 6-7-2 shows the response of the same system for two values of the integration interval: 0.001 and 0.002. In the first case the two methods are identical, in the second the power series produces unreliable results. Because of round off errors originating in the computation of successive powers of matrix $\underline{A}$, it is important to select a proper value for the time interval. Experience gained from the examples included here indicates that the integration interval can

be estimated as:

$$\text{Integration interval DELTH} = \frac{0.1}{\text{largest entry in } \underline{A} \text{ matrix}}$$

$$\simeq \frac{0.1}{\text{largest eigenvalue of matrix } \underline{A}}$$

$$\simeq \frac{0.1}{\text{smallest time constant in the block diagram}}$$

Furthermore the integration interval is particularly important when using the power series method. The Cayley-Hamilton technique, appears to tolerate a time interval ten times larger than recommended above without any appreciable error. In this regard the Cayley-Hamilton technique is superior to the power series method. To complete the comparison between Cayley-Hamilton and power series methods, running times for some of the discussed examples were taken. They are shown in Table 6-3. No conclusion may be drawn as to which method is faster in the range of dimensions of the problems investigated. Running time will increase with the dimension of the problem; a greater difference between the two methods would likely show up for larger matrices.

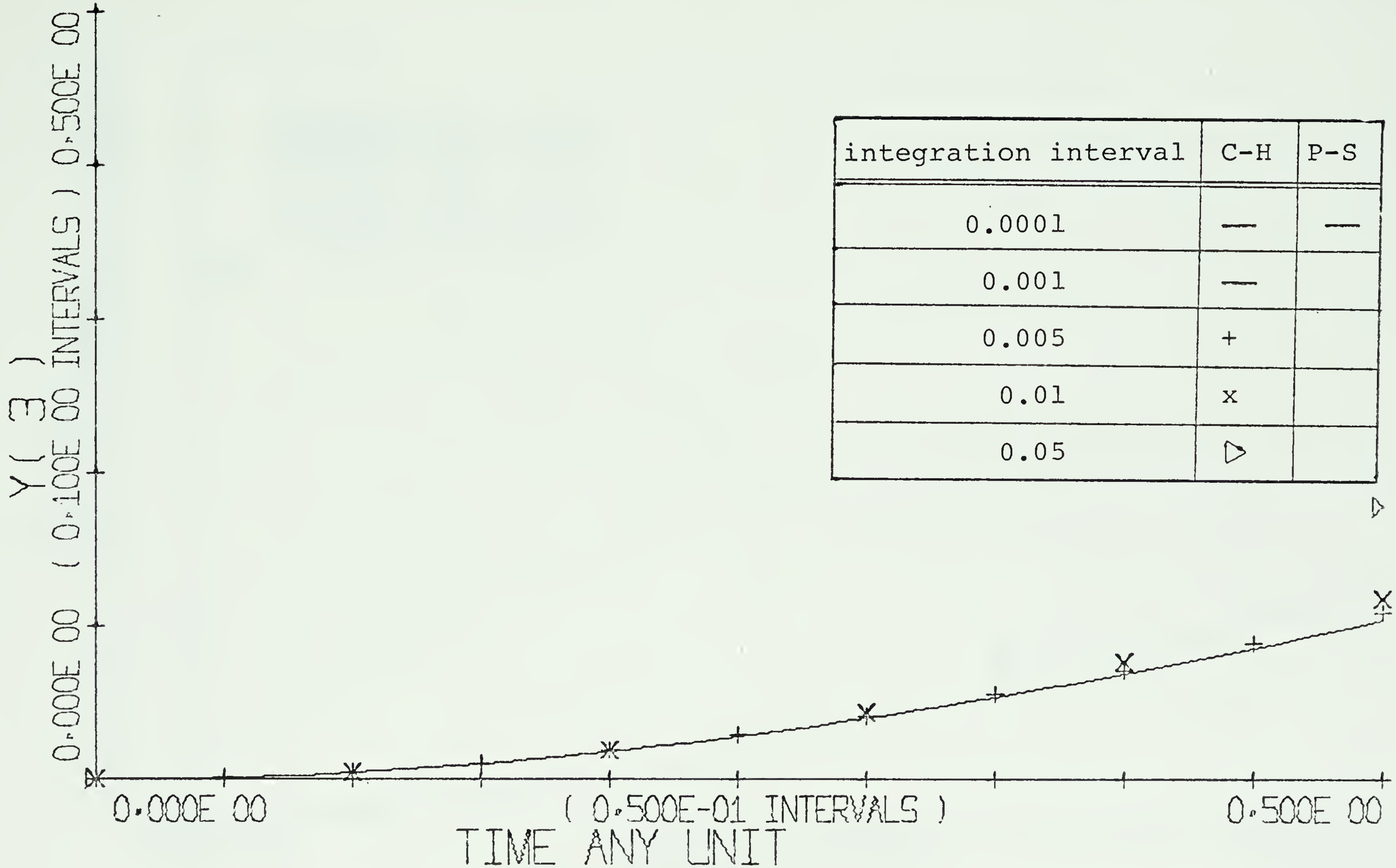


FIGURE 6-7-1 Time Domain Response for Example 6-7: Initial Response

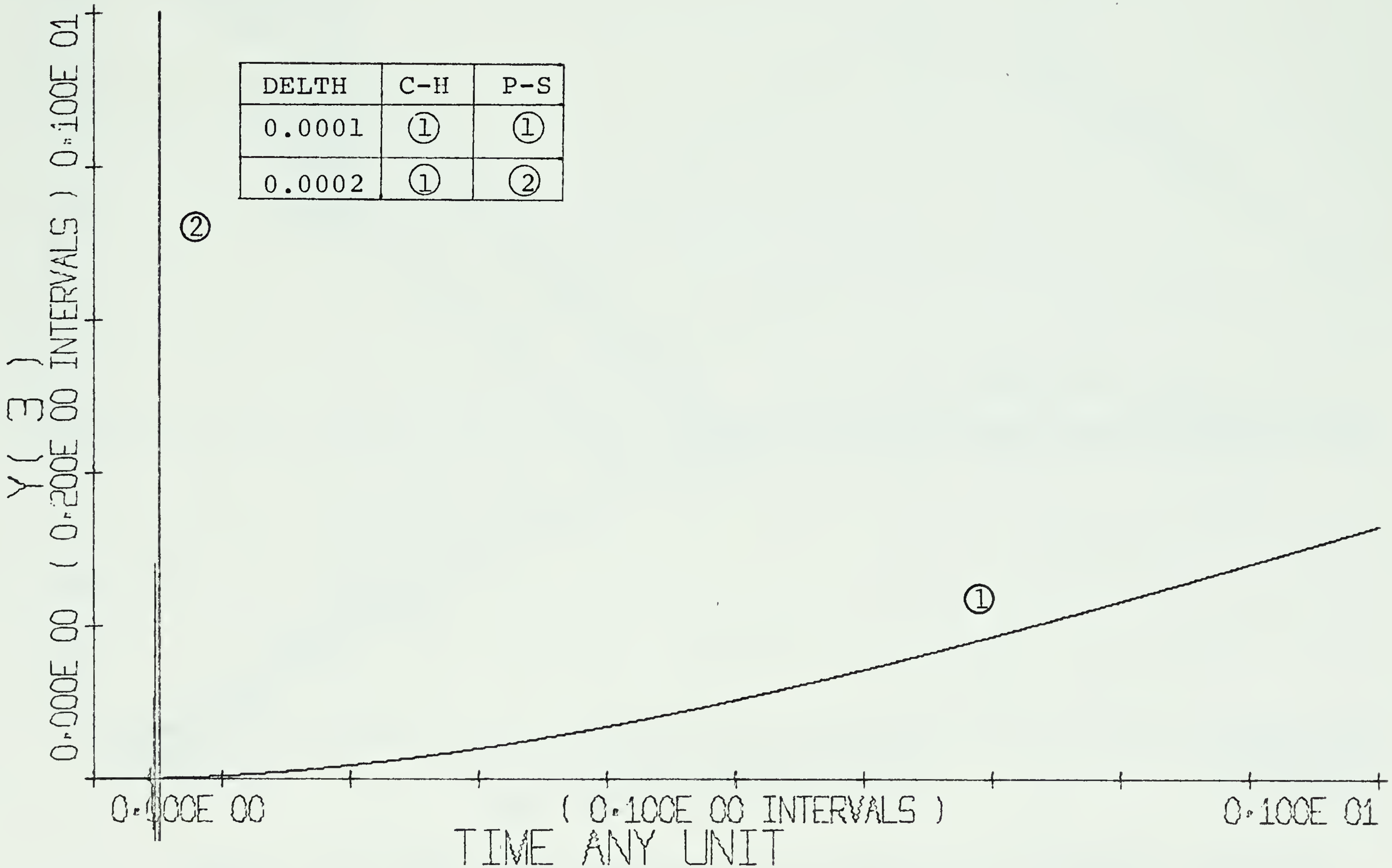


FIGURE 6-7-2 Time Domain Response for Example 6-7: Complete Response

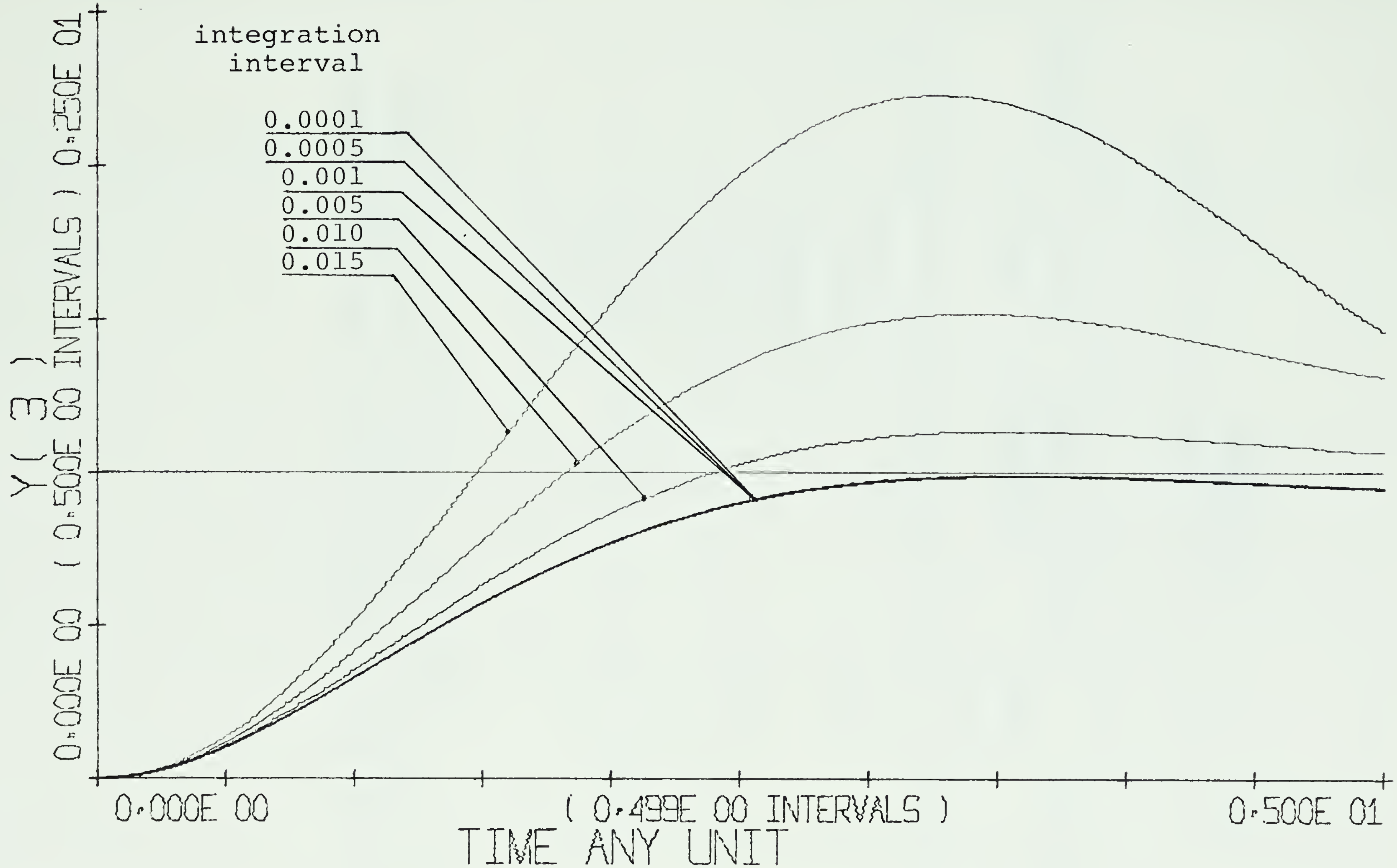


FIGURE 6-7-3 Time Domain Response for Example 6-7: Complete Response for Cayley-Hamilton Technique

TABLE 6-3

Running Time for Examples 6-1,6-2,6-3,6-4 *

Example	Type of Input	Number of State Variables	Time (sec)	
			Power Series	Cayley-Hamilton
6-1	matrix	5	23.2	34.2
6-2	transfer function	2	21.6	21.0
6-3	transfer function	2	30.7	32.0
6-3	transfer function	3	41.8	36.0
6-4	transfer function	3	31.0	27.8

* From "end of input data for job number ...",to time domain response.It includes the time to print out a message for each transfer function of the block diagram.

Chapter VII

PROVISIONS FOR EXTENSION AND FUTURE WORK

Provisions have been made at several points throughout the program for extension or linkage with other programs so as to enhance its present capabilities. In the following paragraphs major extensions are described in more detail. The description follows the order in which such extensions would appear in the detailed flow diagram given on pages B-3 to B-15 of Appendix B.

a. The transfer function and configuration data could be utilized to generate Bode, Nyquist, Nichols plots. All necessary information for this purpose is stored on a disk FILE. For FILE definition and use refer to Tables D-1 and D-2 of Appendix D. The linkage with such an extension is provided by a utility program option value entered by the user (p.B-5).

b. Process transfer function parameters could be found by fitting a first or second order transfer function model to the reaction curve of the real process to a step change of an input stream. Bakke's modified step type of analysis (2) would provide the same information. Process transfer function parameters could be supplied to this program by another program or subroutine using the linkage provided (p.B-6).

This would effectively combine process identification with process simulation and/or control synthesis.

c. The "library of functions to generate controller constants" may be easily increased to supply constants for other types of process transfer functions (p.B-6). Subroutines similar to the one implemented, i.e., GTCC1, could be added. They would use the "LOCAL" function of the IBM TSX operating System (15) so that no additional core storage would be required. If the program is to be extensively used for simulation of an experimental set-up as, for example, the double effect evaporator, the "library" may be extended to accept controller constants estimated by on-line, trial and error, experimental procedures.

d. The system of first order linear differential equations, i.e., the matrix state equation, could be integrated directly by using a Runga-Kutta type of algorithm to generate the time domain response (p.B-10). This would provide a convenient check. The time required for computing large systems(*) by this method, however, has been shown by Davison to be excessively large in comparison with others (7).

e. New sets of subroutines to provide the eigenvalues of coefficient matrix $\underline{A}$ could be easily implemented. The "LOCAL" function previously described would again be recommended to economize on the core storage requirements. Unused values of option number 4

(*) Systems of order 10 or larger

(i.e., NOP(4) = 3,4, etc.) would specify this new set of subroutines (p.B-10).

f. If the eigenvalue finding subroutine(s) provide a non singular matrix of the eigenvectors and this is only possible if the eigenvalues are real and distinct, then the following similarity transformation may be used to find the fundamental matrix

$$\underline{\phi} = e^{\underline{A}} = \underline{M}e^{\underline{\Lambda}}\underline{M}^{-1}$$

where $\underline{\Lambda}$ is the diagonal matrix of the eigenvalues and $\underline{M}$ is the matrix of the eigenvectors (p.B-11).

g. Another extension readily implemented is the evaluation of control criteria such as IAE (integral of absolute error), ISE (integral of squared error), ITAE (integral of absolute error multiplied by time) or other control criteria (1/4 decay ratio, rise time, settling time etc.) from the time domain response. This would allow the adjustment of controller constants to meet the specified control criteria in an off-line type of optimization. Presently the response is observed and controller constants are changed by the operator (see closed-loop tuning, example 6-3). The linkage with a program calculating integral and other control criteria may be obtained, for example, by setting option number 8 to 4,5, etc. This method would imply storing the time domain data for variables of interest

into a permanent FILE to be used later by a program or subroutine that would calculate the control criteria. An off-line program to evaluate control criteria based on a FILE of time domain data is already available in the Data Acquisition, Control and Simulation Center in the Department of Chemical and Petroleum Engineering. Alternatively, the coding to implement evaluation of the criteria could be integrated with the section of programming that calculates the time domain solution.

h. The program was written to interface with two types of optimization programs. In the first case the state difference equation model generated by this program would be transmitted to an optimization program, for example, one using linear programming, which would return the optimum forcing function vector $\underline{U}(t)$. The open loop model response to the specified $\underline{U}(t)$ could then be generated using this program. In the second method the results from one simulation run could be transmitted to an optimization program, for example, one using direct search techniques, which would generate new values of the control parameters.

7.1 Suggestion for Improvements Based Upon User's Experience

The experience of the author and other persons using the program has suggested the following modifications and changes:

a. Start the printed output of the block diagram information and the time domain response on a new page.

b. For the time domain response, have the user supply any two of the following three values instead of DELT and NPTS as at present:

   b.1 DELT = time interval

   b.2 NPTS = number of points

   b.3 TMAX = final time value

c. Build a "permanent" FILE of time domain data of variables of interest so that the data could be used by other programs at some later time.

d. Make the input of coefficients on the block diagram configuration cards optional. The program would then assume that all coefficients were unity unless otherwise specified. For example, with the present FFIOR input subroutine, the user could first enter the input identification number and coefficient for inputs with non unity coefficients followed by a list of the other input identification numbers.

e. Modify the program to select a suitable value of the time subinterval used in the evaluation of the

state difference equation based upon the magnitude of the eigenvalues.

f. Modify the program so that at the completion of a given problem a message is written asking the user which, if any, changes he wishes to make. This would be equivalent to the options presently implemented using coded values of NOP(9) (refer to Appendix A, Table A-2).

g. Rearrange the grouping of the program options

CRIT TO DELT DELTH

and

NPTS

as:

CRIT TO

and

DELTH DELT NPTS TMAX

Then all parameters effectively used for the time domain solution would be grouped in one card (refer to User's Manual, Appendix A).

h. Modify the power series subroutine to use nested multiplications to calculate the powers of a matrix. Execution time and round-off errors resulting from direct multiplication of matrices with large elements are thus reduced. For example, the third element of the power series would be calculated as

$$(((\underline{A}t/1)\underline{A}t/2)\underline{A}t/3) \text{ instead of: } (\underline{A}^2)(\underline{A})\frac{t^3}{3!}$$

7.2 Recommendations

It has been mentioned that changing the scaling of the program from its present dimensions would not present any special difficulties. Due to the small size of variable core, however, it would soon require further subdivisions or "links." Furthermore example 6-7 has pointed out the erroneous results obtained as a consequence of round-off errors and poor accuracy. Consequently, small time intervals had to be used with the obvious inconvenience of considerably increasing the number of iterations, hence the computational time. Extended precision is recommended for those parts of the program dealing with the Cayley-Hamilton technique and power series methods to find the fundamental matrix. As the IBM 1800 FORTRAN IV Language (14) does not permit the mixing of single and extended precision within the same program, a subroutine has been provided by the Data Acquisition and Control System Centre to convert from single to extended precision and vice versa. The computation in extended precision could thus be restricted to those parts, as the one above recommended, that are mostly affected by round-off errors.

Experience to date has shown that the "hands-on", "conversational" mode of operation of the present program is extremely desirable, especially for the control student. However there is a need to get very accurate solutions to much larger problems than the ones envisioned when this program was designed. For these larger problems it would

undoubtedly be desirable to rewrite this program for a larger computer such as the IBM 360/67 available at the University of Alberta Computing Centre. The larger core memory, more powerful FORTRAN language, more extensive subroutine library, faster execution time, longer word length, etc., would all combine to produce a simpler, more accurate and fast program for the handling of off-line batch runs. Future extensions to the time sharing services of the 360/67 might permit interactive use via a typewriter terminal in a manner analogous to the original IBM 1800 program.

A further improvement would be to restructure the program so that those parts dealing with matrix manipulation were programmed in MATLAN or Matrix Language (12). This would considerably reduce the length of the program and likely speed up the overall solution of a problem.

CONCLUSIONS

A computer program has been presented whereby continuous linear control systems represented either in block diagram, transfer function form or, more directly, by a set of first order differential equations in matrix form are analyzed and their time domain response to a set of external forcing functions found.

The examples investigated in Chapter VI and the discussions and comments included at several points throughout this work, confirm that the program has fulfilled the original design specifications outlined in the introductory section, that is:

It is simple and convenient to use but also capable of handling realistic control problems.

It runs in a time sharing mode on a small real-time computer.

It generates the equivalent "state variable" matrix representation for a problem defined in block diagram notation with transfer functions given in Laplace transform notation.

It generates the time domain solution from the matrix difference equations.

It generates the time domain solution for all elements of the state variable vector $\underline{X}$ or the "output"

variable vector $\underline{Y}$ either as listings, graphs produced on a digital plotter or displays on a storage oscilloscope.

It has provision for linkage to programs for closed-loop optimization of controller constants.

It has provision for linkage to alternative subprograms for all internal steps such as determining the eigenvalues, etc.

It handles control systems whose characteristic equations have real, complex and/or repeated roots.

It produces complete problem documentation of both the control results and the important steps of the mathematical solution.

It accepts forcing functions defined mathematically or simply as a list of function values.

It does not usually require any block diagram manipulation or standardization by the user prior to using the program.

It handles "standard" control problems by special options.

It handles pure time delays, time delays associated with transfer functions and isolated derivative actions.

Up to two sampler plus zero-order elements can be handled [equation (4-1)].

It has user's intervention capabilities in a "conversational" mode, analogous to other simulation programs, which permit the user to alter program specifications and parameters and rerun the problem.

The six examples presented and the use by other students confirm the usefulness of the program to solve realistic control problems.

The control oriented block diagram, transfer function input feature, simplified to the extent of requiring no block diagram manipulation and minimum input data, should encourage process control students to use the program to carry out simulation studies and to gain experience with control techniques.

BIBLIOGRAPHY

1. Anderson, N. A., "Control Modes by Step Analysis", Instruments and Control Systems, p. 113, December, 1963.

2. Bakke, R. M., "A Straightforward Approach Allowing Industrial Operators to Adjust Continuous and Digital Feedback Controllers with the Aid of a Digital Computer", Preprints, IFAC/IFIP Symposium, Toronto, June 17-19, 1968.

3. Brennan, R. D., Silberberg, M. Y., "Two Continuous System Modeling Programs", I.B.M. System Journal, Vol. 6, No. 4, p. 242 (1967).

4. Buckley, P. S., "Controller Settings and Loop Performance - A Rapid Estimate", Control Engineering, p. 81, August 1964.

5. Chien, K. L., Hrones, J. A., Reswick, J. G., "On the Automatic Control of Generalized Passive Systems", Transactions of the ASME, Vol. 74, No. 2, pp. 175-178, February, 1952.

6. Coughanowr, D. R., Koppel, L. B., "Process System Analysis and Control", McGraw-Hill (1965).

7. Davison, E. J., "The Numerical Solution of Large Systems of Linear Differential Equations", A. I. Ch.E. Journal, Vol. 14, No. 1, pp. 46-49, January, 1968.

8. DeRusso, P. M., Roy, R. J., Close, C. M., "State Variables for Engineers", Wiley, New York (1965).

9. Franks, R. G. E., "Mathematical Modeling in Chemical Engineering", Wiley, New York (1966).

10. Gallier, P. W., Otto, R. E., "Self-Tuning Computer Adapts DDC Algorithms", Instrumentation Technology, p. 65, February, 1968.

11. I.B.M., 1130 Continuous System Modeling Program", (1130-CX-13X), Program Reference Manual (H20-0282-0).

12. I.B.M., "System/360 Matrix Language (Matlan)", Program Description Manual (H20-0564).

13. I.B.M., "System/360 Scientific Subroutine Package (360A-CM-03X) Version III, Programmer's Manual (H20-0205-3).

14. I.B.M. 1130/1800 Basic FORTRAN IV Language, File No. 1130/1800-25, Form C26-3715-3.

15. I.B.M. 1800 Time-Sharing Executive System Operating Procedures, Program Number 1800-0S-001, Version 3, File No. 1800-36, Form C26-3754-3.

16. Kalman, R. C., "Design of a Self-Optimizing Control System", Transactions of the ASME, pp. 468-478, Febr February, 1958.

17. Koepcke, R. W., "A Discrete Design Method for Digital Control", Control Engineering, pp. 83-89, June, 1966.

18. Lopez, A. M., Murrill,P. W., Smith, C. L., "Optimal Tuning of Proportional Digital Controllers", Instruments & Control Systems, October, 1968.

19. Lopez, A. M., Murrill,P. W., Smith, C. L., "Tuning PI and PID Digital Controllers", Instruments & Control Systems, February, 1969.

20. Miller, J. A., Lopez., A. M., Smith, C. L., Murrill, P. W., "A Comparison of Controller Tuning Techniques", Control Engineering, December, 1967.

21. Murrill,P. W., Smith, C. L., "Controllers, Set Them Right", Hydrocarbon Processing, Vol. 45, No. 2, February 1966.

22. Ogata, K., "State Space Analysis of Control Systems", Prentice-Hall, Englewood Cliffs, N. J., (1967).

23. Saucedo, R., Schiring, E. E., "Introduction to Continuous and Digital Control Systems", MacMillan, New York (1968).

24. Thayer, D., Cohen, E. M., "Tuning Direct Digital Control", Instruments and Control Systems, p. 85, October, 1967.

25. van der Grinten, P.M.E.M., "Finding Optimum Controller Settings", Control Engineering, pp. 51-56, December, 1963.

NOMENCLATURE

G_i A general transfer function

G_v Transfer function of a value

G_m Transfer function of a measuring element

K_p Controller gain

τ_I Integral time parameter for controller

τ_D Derivative time parameter for controller

τ_d Time delay

τ Process lag

CONTROL ALGORITHMS

P Proportional controller

PI Proportional + integral controller

PID Proportional + integral + derivative controller

CONTROL CRITERIA

3-C Three constraints [reference (20)]

IAE Integral of the absolute error

ISE Integral of the square error

ITAE Integral of the time multiplied by the absolute error.

Note that the following methods of identifying the elements of vectors, are equivalent:

example: $y_3(t) = y_3 = Y(3)$

APPENDIX A

User's Manual

Table Number

Figure Number

A.1 General Considerations

The program is designed to accept input in block diagram, transfer function (TF) form and produce the equivalent state equation in the form:

$$\dot{\underline{X}} = \underline{A}*\underline{X} + \underline{B}*\underline{U}$$
$$\underline{Y} = \underline{C}*\underline{X} + \underline{D}*\underline{V} \tag{A-1}$$

If the input is in matrix form the above system is supplied by the user. The program then calculates the eigenvalues of the coefficient matrix $\underline{A}$ finds the fundamental matrix $\underline{\phi}(T)$ by typically using the Cayley-Hamilton technique and produces the coefficient matrix $\underline{H}(T)$ of the state difference equation

$$\underline{X}_{i+1} = \underline{\phi}(T)*\underline{X}_i + \underline{H}(T)*\underline{U}_i$$
$$\underline{Y}_i = \underline{C}*\underline{X}_i + \underline{D}*\underline{V}_i \tag{A-2}$$

where time = iT and $\underline{H}(T)$ is the integral between zero and T of $\underline{\phi}(t)*dt*\underline{B}$. It finally calculates the time domain response by iterating equation (A-2) over a specified number of time intervals.

The system is so designed as to keep the amount of input data to a minimum. Non essential data are initialized by the program to "standard values" thus simplifying the specification of a problem. What is meant by "standard values" and how other than standard values may be entered will be discussed later on.

The program was designed to be as free as possible of arbitrary input formats and rigid sequences that the user must follow. The following sections describe a systematic approach to the use of the program that is recommended for the beginner. The experienced user will realize that there are several alternative means of specifying the same problem and can use the approach best suited for his problem. The fully documented examples which occur at the end of this manual illustrate a number of the alternative methods and options.

A.1.1 Problems Specified by a Block Diagram and Transfer Functions

It is assumed that the user has drawn a block diagram that adequately defines the problem he wishes to solve. There is no standard format that the user must follow and he can specify multivariable, multiple path systems if he wishes.

It is worthwhile mentioning that a block diagram can usually be entered in its original form, thus requiring no user's manipulation, or can be modified into an equivalent, more compact one. This requires user's time but saves on the overall dimensions of the problem, and hence computer time. The time domain results should be identical. A simple block diagram manipulation is shown in Figure A-1.

The user should number every block in his diagram, including any summing points, etc., starting with one and

proceeding with successive integer values. The starting point and order of the numbering is not important. By convention each element in a block diagram can have only one output variable which is assumed to be identified by Y(i) and the program sets "i" equal to the block from which the signal originates, i.e., the output from block three is referred to as output variable Y(3).

External functions such as setpoints, loads, disturbances, etc., are designated as elements of the vector $\underline{u}$. The user should number each such function u(101), u(102), u(103)... Again the order is unimportant.

The user should note that the existing IBM 1800 version of the program is limited by DIMENSION statements to the following: Maximum number of state variables in $\underline{X}$ is 10.

Maximum number of output variables in $\underline{Y}$ is 15

Maximum number of forcing functions in $\underline{u}$ is 10.

After completing these steps the user is ready to prepare the "configuration" cards which are used to input the block diagram information to the program.

However before specifying how to prepare input cards it is helpful to know some of the general characteristics of the free format input routine used by this program.

A.1.2 Free Format Input Subroutine

Data are supplied in free format being read in by a special FFIOR subroutine.(*) The length of the input list read by this subroutine is arbitrary but at present the

---

(*) see page F-1 of Appendix F.

maximum number of integer, real and alphameric values read in by a single call statement is limited to a maximum of 50 real and integer values, or 100 alphamerics.

Data may be entered separated either by a comma or by one or more blanks. Alphamerics should always be entered between single asterisks. The end of an input list is given by a double asterisk preceded and followed by a blank. Care must be exercised to enter real values with their decimal points or with an Ew.d. format. As the number of real, integer and alphameric values are not specified, no error message is caused by entering a real value without its decimal point. The subroutine, however, interprets that value as an integer and serious errors will undoubtedly follow. Similarly the FFIOR subroutine cannot usually determine whether one or more values are omitted from any particular list but the rest of the program will not execute properly without them.

Integer and real numbers can be intermixed on the same card or input record. One of the first operations performed by the FFIOR routine is to sort the input data into separate vectors of integer, real and alphanumeric data. The data in the input list of the calling program is then taken from these storage vectors in the order they were stored. Therefore integer and real numbers can be intermixed in any fashion as long as the relative order of the integers to one another and the real numbers to one another is maintained. For example the following are all equivalent:

1,1.0,2,2.0b3b3.0 **

1,2,3,1.0,2.0,3.0 **

1.0b2.0b1b2b3b3.0 **

but 3,2,1,1.0,2.0,3.0 ** would reverse the order of the first and third integer.

If real and integer values are related as for example output variable 3 and its coefficient 1.0 then they must have the same order in the data list, for example, they both must be the 2nd integer and the 2nd real value. If the user wishes to identify his output by means of a title he may do so by punching it on one or more cards. Each card must have a single asterisk ahead and after the alphameric field. Blank cards may be used to space printed lines.

A.2 First Data Card

Two types of input are possible according to the value of the input matrix flag (MFLAG).

MFLAG = 0 implies a TF and configuration data input

MFLAG = 1 implies a matrix input.

Matrix input is described in a later section of this manual. For the case of block diagram/transfer function input the first card should be:

0 **

A.3 Transfer Function Specification

The block diagram configuration of a typical problem consists of several blocks relating one or more inputs to a single output through an expression or, transfer function,

identified by the same integer constant as the block.

In general a transfer function is the ratio of two polynomials in "s", the degree of the polynomial in the numerator being less or equal to that in the denominator. However, a transfer function can also be a constant, a pure time delay, etc...

A.3.1 Polynomial Terms

A general form for such type of transfer function is

$$G(s) = \frac{(a_1s+a_2)(a_3s^2+a_4s+a_5)(a_6s^3+a_7s^2+\ldots)(\ldots)\ldots}{(b_1s+b_2)(b_3s^2+b_4s+b_5)(b_6s^3+b_7s^2+b_8s+b_9)(\ldots)\ldots} \qquad \text{(A-3)}$$

Transfer function (A-3) is identified by the integer number assigned to the particular element of the block diagram and each term of the numerator and denominator is identified by two factors:

1) an integer giving the degree of the term (a preceeding negative sign is used to indicate terms in the denominator).
2) a series of coefficients in number equal to the degree plus one.

A.3.2 Time Delay Terms

A pure time delay $e^{-\tau_d s}$ has:

1) degree = 0 (by definition)
2) coefficients 1., τ_d

A.3.3 Constant Terms

A constant k is considered as a special case of the "zero order" transfer function previously described:

$$ke^{-\tau_d s} = ke^0 = k$$

therefore

1) degree=0

2) coefficients k,0.0 (in this order)

It should be noted that a constant term cannot be entered as a special case of a polynomial by defining all powers of "s" to be zero. In most cases, however, a constant gain factor could be entered as a coefficient associated with the inputs specified in the configuration card for that block (see example A-5-1). When a zero order term is present, it must be in the numerator of the transfer function and must be the first one in the list of numerator terms. The order in which the other numerator and denominator terms are given is irrelevant.

A.3.4 Transfer Function Input

With the above specifications, the input for the transfer function

$$G(ID) = \frac{ke^{-\tau_d s}(a_1 s+a_2)(a_3 s^2+a_4 s+a_5)(\ldots)..}{(b_1 s+b_2)(b_3 s^2+b_4 s+b_5)(b_6 s^3+\ldots)(\ldots)..} \quad \text{(A-4)}^{(*)}$$

takes the general form:

(*) A sampler plus zero-order-hold element, $(1-e^{-Ts})/s$, can be specified using equation (A-4).

$$ID,0,k,\tau_d,+1,a_2,a_1,+2,a_5,a_4a_3,+3,\ldots,\ldots$$

$$-1,b_2,b_1,-2,b_5,b_4,b_3,-3,\ldots,\ldots,**$$

where:

ID is the block, transfer function, number assigned by the user.

$0,k,\tau_d$ represent a time delay term as previously discussed and

each term is entered with a signed integer which gives the degree of that term (+ for numerator terms; - for denominator terms) followed by a list of coefficients (all powers must be present) ordered from low to high powers.

Note that in the above general input expressions:

ID, 0, +1, +2, ... -1, -2, -3, ... must be entered as integer values,

k, τ_d, a_1, a_2, ... b_1, b_2 ... must be entered as real values with decimal point or in E format.

Example A-3-4-1

$$G(5) = \frac{3e^{-5s} \quad (s+1)}{5s^2 + 27s + 9}$$

INPUT LIST: 5, 0, 3., + 5., +1, 1.0, 1.0, -2, 9., 27., 5.,**

Example A-3-4-2

$$G(4) = \frac{(s+1)}{(s+6) \ (s^2+36s+4)}$$

INPUT LIST: 4, +1, 1., 1., -1, 6., 1., -2, 4., 36., 1., **

Although the program processes almost any type of block diagram, hence transfer functions, the following limitations apply:

(i) The program handles only one isolated (pure) time delay.

(ii) Up to two time delays are presently allowed if each of them is associated with a transfer function for which the degree of the numerator is less than the degree of the denominator.

(iii) When an isolated derivative action is present (s + a) no time delays are allowed except an isolated one. Note that the derivative term must be of 1st order and so located in the block diagram as not to cause any derivative of external forcing functions such as set points, load disturbances.

(iv) If the degree of the numerator and denominator of the block diagram as a whole (i.e., the sum of the highest powers of the numerator and denominator terms of all transfer functions constituting the diagram) are equal, then no time delay is allowed. Note that this rule also applies to single transfer functions.

The order in which transfer functions and their configuration data (see section A.5) are entered is irrelevant.
The only exception occurs when the system is requested to provide controller constants stored in a "library". In

such case the process transfer function must precede the controller transfer function in the card deck since the process constants are used in the evaluation of the controller constants (see section A.4.6).

A.4 Special Transfer Function Input

Special coding may be used to speed up the input of frequently occurring types of transfer functions. These special cases are best illustrated by examples.

A.4.1 First Order Transfer Function with Time Delay

$$G(6) = \frac{ke^{-\tau_d s}}{\tau_I s + 1}$$

a. standard input: $6,0,k,\tau_d,-1,1.,\tau_1,**$

b. special input: $206,k,\tau_d,\tau_I,$ **

where the ID number is formed by adding 200 to the transfer function or block number 6.

A.4.2 Second Order Transfer Function with Time Delay

$$G(3) = \frac{ke^{-\tau_d s}}{(\tau_1 s+1)(\tau_2 s+1)}$$

a. standard input: $3,0,k,\tau_d,-1,1.0,\tau_1,-1,1.0,\tau_2,**$

b. special input: $303,k,\tau_d,\tau_1,\tau_2,**$

ID is 300 + transfer function number.

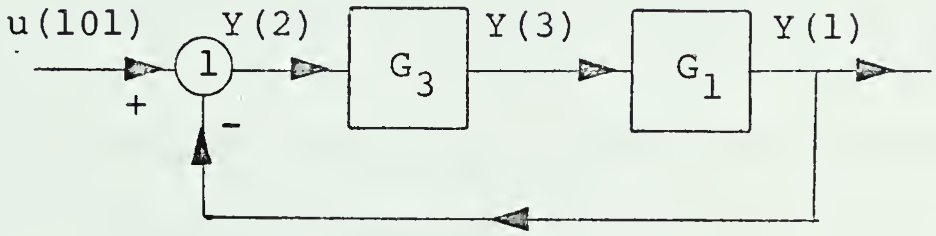
A.4.3 Proportional Controller Transfer Function

G(2) = Kc

a. standard input: 2,0,Kc,0.0,**

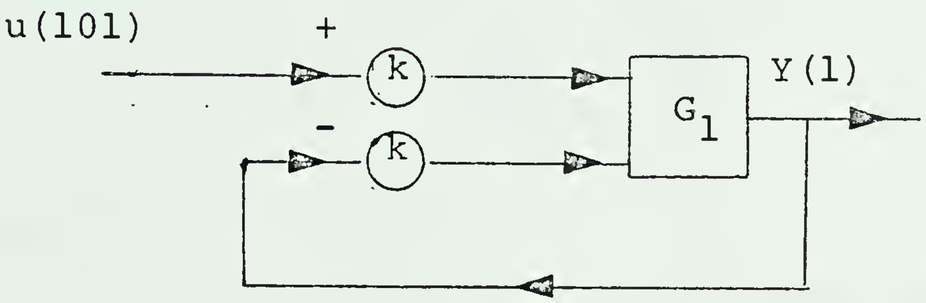
b. special input : 402,Kc,**

where the ID number is 400+transfer function number.



a)

assuming $G_3 = k$



b)

	Configuration Data	Transfer Function Data
a)	101,3,1.0, **	1,as required, **
	103,2,1.0, **	2,0,1.0,0.0, **
	102,1,-1.0,101,1.,**	3,0,k,0.0, **
b)	101,1,-k,101,k, **	1,as required, **

FIGURE A-1 Simple Block Diagram Manipulation and Resulting Simplification of Problem Input

A.4.4 Proportional + Integral Controller Transfer Function

$$G(8) = Kc(1 + 1/\tau_I s) \quad \text{or}$$

$$G(8) = \frac{Kc\tau_I s + Kc}{\tau_I s + 0}$$

a. standard input 8, +1, Kc, $Kc\tau_I$, -1, 0., τ_I **

b. special input 508, Kc, τ_I **

ID is 500 + transfer function number.

A.4.5 Proportional+Integral+Derivative Controller Transfer Function

$$G(1) = Kc \frac{(1+\tau_D s+\frac{1}{\tau_I s})}{(1+\tau_D/_\gamma s + \frac{1}{\alpha\tau_I s})} = \frac{Kc + Kc\ \tau_I s + Kc\ \tau_I\tau_D s^2}{\frac{\tau_I\tau_D}{\gamma}s^2 + \tau_I s + \frac{1}{\alpha}}$$

Selecting $\alpha = \gamma = 10$

a. standard input 1, +2, Kc, Kc τ_I, Kc $\tau_I\tau_D$, -2,

0.1, τ_I, 0.1 $\tau_I\tau_D$ **

b. special input 601, Kc, τ_I, τ_D **

Note that τ_I, τ_D cannot be set to zero. This would make the coefficient of the largest term in the denominator zero which is not valid.

A.4.6 System Supplied Controller Constants

Once the user has defined his process system he must usually choose a controller and the controller parameters.

For the general problem this could be very difficult. However if the process can be approximated by the following transfer function:

$$Gp(s) = \frac{ke^{-\tau_d s}}{\tau s + 1} \tag{A-5}$$

and if the user wants to employ a single variable unity feedback control system then the program will assist him with the choice of control constants. Specifically, the user has the option of either entering the controller constants in one of the ways previously described or having the system determine the optimum set of a proportional, proportional+integral, or proportional+integral+derivative controller constants based on the following criteria:

criterion number	criterion(*)
1	Zeigler-Nichols
2	3-C
3	IAE
4	ISE
5	ITAE
6	Cohen-Coon
7	Quickest response + 20% overshoot
8	Quickest response without overshoot

(*) These control criteria apply to process transfer functions of the type (A-5) for the following range of process parameters:

Criteria 1 through 6

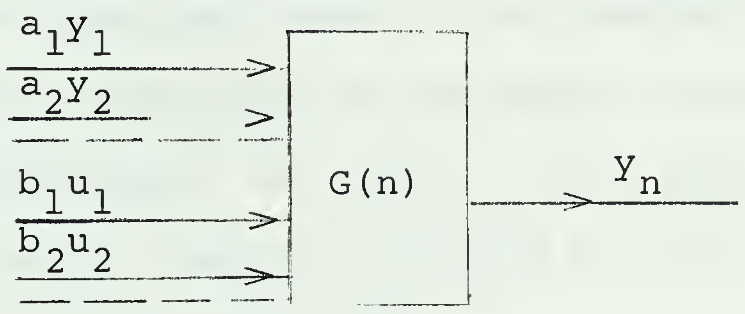
$0 \leq \tau_d/\tau \leq 1.0$ [reference (20)].

Criteria 7 and 8

$0.1 \leq \tau_d/\tau \leq 1.0$ [reference (5)].

A.5 Configuration Data Input

Each transfer function is coupled with its configuration data that define number, type and value of the input(s) to the transfer function, the output always being a single variable y (output variable) identified by the same number as the block or transfer function. A general form for any element in a block diagram is shown below where G(n) is the transfer function:



and the corresponding configuration data card is:

100 + n, 1, a_1, 2, a_2, 3, a_3, ... 101, b_1, 102, b_2 **

where

100 + n is the code of a configuration data input for block "n".

1, 2, 3, identify the three output variables y_1, y_2, y_3 as inputs to block "n".

a_1, a_2, a_3 are the coefficients of y_1, y_2, y_3 respectively. (These must be real constants).

101, 102 identify external forcing functions u_1, u_2 as inputs to block "n".

b_1, b_2 are the coefficients of u_1,u_2. (These must be real constants).

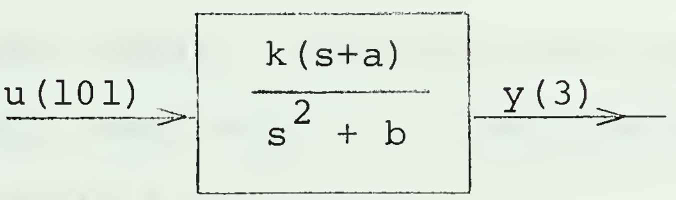
As mentioned earlier because of the way the FFINP input subroutine handles data the same configuration data may be entered as

100 + n, 1, 2, 3, 101, 102, ... a_1, a_2, a_3, b_1, b_2, ** or
100 + n, 1, 2, 3, a_1, a_2, a_3, 101, 102, b_1, b_2, **
etc.

Note that in each case the order of the integers and the orders of the real variables, is the same. In many problems the coefficients "a_i" and "b_i" are equal to ± 1.0.

One configuration card is required for each of the "n" elements in the block diagram. Once the user has defined the configuration he can then specify the transfer function associated with each block.

Example A.5.1



configuration card: 103,101,1.0,**

transfer function input card: 3,0,k,0.0, 1,a,1.0.-2,b,0.0, 1.0,** or

configuration card: 103,101,k,**

transfer function input card: 3,+1,a,1.0,-2,b,0.0,1.0,**

A.6 External Forcing Function or Disturbances

If a time domain response of the system subject to external forcing functions is desired, the type and magnitude of the forcing functions must be supplied. If no such information is given the program stops after having calculated the coefficient matrices of the state equation (A-1).

External forcing functions (set points, loads, etc.) are characterized by the small letter "u" indexed sequentially from 1 up.

Again the disturbance u_1 can be any one of them. A typical external forcing function entry shows as:

999, 100 + ID, disturbance code number, 1st disturbance parameter, 2nd disturbance parameter**.

where

999 is the code used to input disturbances. Only one disturbance may be entered for each 999 card.

100 + ID is the disturbance number (ID is the index of the actual disturbance).

Disturbance code number: specifies the type of disturbance i.e., step, ramp, etc., ... Disturbance code table is given as Table A-1.

Two real value parameters define completely the magnitude of the disturbance (see Table A-1).

Ex. 999, 102, 1, 10.0, 0.00 **

means a step change of magnitude 10 in external forcing function u_2.

TABLE A-1

Disturbance Code

Type	Code Number	First Disturbance Parameter	Second Disturbance Parameter
step	1	magnitude	---
ramp	2	slope	---
sine wave	3	amplitude	radiant frequency
pulse	4	magnitude	duration
-----	5--9 vacant		
file of function data sup-plied by user(*)	10	---	---

(*) to be implemented

The present program can only handle step disturbances that are specified around non-zero initial conditions. Other steady state variables may be entered as step disturbances of magnitude equal to the steady state value.

For the case of "matrix input" (refer to section A.7) it would still be possible to deal with disturbances operating around non-zero steady state values by:

- assigning two variables to each disturbance, the first to be a deviation variable or operating around zero initial conditions and the second to represent the steady state value.
- modifying the matrix $\underline{B}$ and vector $\underline{U}$ in equation (A-1) as:

$$\left[\, B \;\vdots\; B \,\right] \begin{bmatrix} \underline{U} \\ \hline \underline{U}^{*} \end{bmatrix}$$

where $\underline{U}^{*}$ contains the steady state values for variables as described.

A.7 Problem Specified by Matrix Equations

The user has the option of specifying the problem in either:

1) block diagram/transfer function notation
2) matrix equations

If he has selected the first option, he can skip to section A.8. This section describes the input data required for the second option.

A.7.1 First Card for Matrix Input Option

The first input specification card should be:

1,**

in order to identify the following cards as matrix input.

A.7.2 System Equation Matrices

The user must identify his problem with one of the following

$$\dot{\underline{X}} = \underline{\underline{A}}\underline{X} + \underline{\underline{B}}\underline{U} \qquad \underline{Y} = \underline{\underline{C}}\underline{X} + \underline{\underline{D}}\underline{V} \qquad \text{a)}$$

$$\underline{X} = \underline{\underline{A}}\underline{X} + \underline{\underline{B}}\underline{U} \qquad \underline{Y} = \underline{\underline{C}}\underline{X} \qquad \text{b)}$$

$$\dot{\underline{X}} = \underline{\underline{A}}\underline{X} + \underline{\underline{B}}\underline{U}$$

for case b) correct program control entry (see section A.10.1) option 7 to:

0 0 1 7 1 **

for use c) correct same option to:

0 0 1 7 0 **

If no time delay in present the forcing vectors $\underline{U}$ and $\underline{V}$ are assumed to be identical. They contain all forcing functions whose type and magnitude have been previously entered in the format described in section A.6.

If the logical unit numbers for I/O device are those of a 1816 typewriter the user is assisted from now on by proper messages.

Otherwise he must supply:

1) N, M, NF **

where N = dimension of vector X

M = dimension of vector Y

NF = dimension of vector $\underline{U}$ = $\underline{V}$ if no time delay is present. It can be omitted otherwise.

2) NTD **

NTD = number of time delays

If NTD > 0 proceed otherwise jump to 7)

3) $\underline{MV}$ **

This vector contains the number of disturbances and as many pairs of numbers of delayed disturbances and delayed state variables as there are time delays, length therefore is N1 = 2*NTD+1.

4) $\underline{NV}$, $\underline{UD}$ **

Vector $\underline{NV}$ contains the variable number for variables such as $\underline{u}$, $\underline{u}_D$, $\underline{X}_D$ entered in $\underline{MV}$ vector. $\underline{UD}$ contains values of time delays associated with variables contained in $\underline{NV}$ vector. Note that if a variable, for example u_3, is not delayed its corresponding entry in $\underline{UD}$ is 0.0 length of $\underline{NV}$ and $\underline{UD}$ vectors is N2. For cases 2) and 3) jump to 7).

5) $\underline{MV1}$ **

This vector is analogous to $\underline{MV}$ with reference to forcing vector $\underline{V}$ instead of $\underline{U}$, length is N3 = 2*NTD+1.

6) $\underline{NV1}$, $\underline{UD1}$ **

These two vectors are analogous to $\underline{NV}$ and $\underline{UD}$. They

refer to above defined vector MV1; length is N4.

7) A **

Enter matrix A columnwise (N*N elements). Note that up to 50 elements may be entered in a single data field on as many cards as necessary. However any number of elements (less than 50) may be entered at the time followed by ** as long as the columnwise sequence is not discontinued. The same considerations apply to matrices B, C, D.

8) Enter matrix B columnwise.

Elements are N*NF if NTD = 0

N*N2 if NTD > 0

For use c) stop here

9) Enter matrix C columnwise

Elements are M*N

For case b) stop here

10) Give matrix D columnwise

Elements are M*NF if NTD = 0

M*N4 if NTD > 0

A.8 Last Entry

It is either 999 0 ** (Continue)

999 1 ** (Stop)

The second integer gives the value of a non execution flag. In the first case the program is continued in the second stopped. Note that if the card previous to this one specified the logical unit numbers for input/output to be those of the 1816 typewriter terminal then this entry must

be made from the keyboard.

A.9 Input Card Deck

In order to start the program it is necessary to initiate a job by placing the following control cards valid only for the IBM 1800 TSX system and data cards in the input hopper of the card reader:

```
   4        10 15 20        40
// JOB      X  X  X         NAME

   4   8        16
// XEQ LAST1    FX
```

INPUT DATA CARDS

0,** block diagram input specification

100+n....** configuration cards

n....** transfer function card

999....** forcing function specification card

0,0,....** special program parameters (see page A-37)

999,0, ** end of input data and execution

// JOB

// END OF ALL JOBS

BLANK CARD

Note that as an extreme case the input data could be given by a single card specifying the logical unit number for the 1816 terminal. The user would have to enter all the remaining data via keyboard.

For example, consider the following system defined in matrix form:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & 4 & 5 & 6 \\ 2 & 0 & 4 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_{2D} \\ x_{1D} \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 6 & 2 \\ 1 & 3 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & 6 & 1 \\ 0 & 2 & 3 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_{2D} \\ x_{1D} \end{bmatrix}$$

with reference to entries 1) to 10) described in section A.7.2, the user should enter:

1) 2,3,2,**
2) 1,**
3) 2,1,1,**
4) 1,2,2,1,0.0,0.0,0.3,0.3,** where 0.3 is the time delay associated with u_{2D}, x_{1D}
5) 1,1,1,**
6) 1,2,1,0.0,0.3,0.3,**
7) 1.0,3.0,2.0,4.0,**
8) 0.0,2.0,4.0,0.0,5.0,4.0,6.0,1.0,**
9) 6.0,1.0,0.0,2.0,3.0,4.0,**
10) 0.0,0.0,1.0,6.0,2.0,1.0,1.0,3.0,2.0,**

A.10 Program Control Entries

The user's problem is completely specified by the:

1) configuration and transfer function data (or the equivalent matrix input data).

2) external forcing function specifications.

The program is set up so that it will accept this input and produce a "standard" solution which includes a print out of all the output wariables, $\underline{Y}$, at the time intervals specified. If this is sufficient for the user then no further specifications are required.

However if the user wishes to specify which mathematical techniques are used during the solution and/or to specify for example that certain elements of $\underline{Y}$ should be plotted on the digital plotter then he must supply additional specifications to the program.

All the presently available options are specified in the following sections. The user may elect to change none, a selected few, or all of the options from their standard values.

A.10.1 Vector of Options

Most of the options defining the steps to be followed in the solution of any problem are defined by setting the elements of the vector NOP(i) to the desired integer value shown in Table A-2. The user is referred to the more extensive description of the program given in Chapters II, IV and V for details of the methods.

TABLE A-2

Vector of Options

NOP(1)	= 0	transfer function plus configuration data input
	= 1	matrix input
NOP(2)	= 0*	standard integration of matrix equation
	= 1†	Runga-Kutta method for integration
NOP(3)	= 0*	Cayley-Hamilton technique to find the fundamental matrix
	= 1	power series method
NOP(4)	= 1	CHEQN+BART subroutines to find eigenvalues of matrix $\underline{A}$
	= 2*	CHEQN+PRBM (SSP)
	= 3	other
	= 4	other
NOP(5)	= 0*	standard procedure to find the fundamental matrix
	= 1†	use eigenvectors to find the fundamental matrix
NOP(6)	= 0*	stop for positive eigenvalue
	= 1	proceed
NOP(7)	= 0	both matrix $\underline{C}$ and $\underline{D}$ are ignored
	= 1	only matrix $\underline{D}$ is ignored
	= 2*	neither $\underline{C}$ nor $\underline{D}$ is neglected
NOP(8)	= 0	no action taken
	= 1*	print data as they are generated
	= 2	store data on disk file to plot on scope
	= 3	store data on disk file to plot on plotter
	= 4†	store data on permanent file for later use
NOP(9)	= 0	no changes desired (exit)
	= 1	change disturbance parameter or type
	= 2	change option value,etc..
	= 3	change transfer function type and/or parameter
	= 4	vacant
NOP(10)	=	vacant

* = standard option

† = method not implemented in present program

NOP(1) is set by the first data card as described in section A.2.

NOP(2) The program presently calculates the fundamental matrix in order to define a recursive difference equation which will give the discrete time domain response $\underline{Y}(n\underline{T})$, $\underline{X}(n\underline{T})$. NOP(2) = 1 is a provision for future addition of an alternative method such as Runga-Kutta integration.

NOP(3) Permits the user to choose one of (at present) two alternative methods of finding the fundamental matrix. The methods are independent and offer a good means of checking the accuracy of the program. Under some conditions one method will work when the other fails.

NOP(4) Two alternative implementations of the same method of calculating the distinct, and/or repeated and/or complex eigenvalues of $\underline{A}$ are provided. Others may be readily added.

NOP(5) Is a provision for future addition of a method of calculating the fundamental matrix $\underline{\phi}(T)$ using the eigenvectors. This option improves the execution time when all eigenvalues are distinct.

NOP(6) Positive eigenvalues mean an unstable time domain solution and could result from, say, a poor choice of controller constants. NOP(6) = 0 will stop the solution at this point and inform the user rather than proceed with the solution.

NOP(7) permits three different formulations of the problem for matrix input.

For $\dot{\underline{X}} = \underline{A}\underline{X} + \underline{B}\underline{U}$

$\underline{Y} = \underline{C}\underline{X} + \underline{D}\underline{V}$ NOP(7) = 2

or $\underline{Y} = \underline{C}\underline{X}$ NOP(7) = 1

or $\underline{Y}$ is not calculated NOP(7) = 0

NOP(8) defines what is to be done with the time domain data $\underline{X}(nT)$, $\underline{Y}(nT)$.

NOP(8) = 0 is an option to omit the calculation.

NOP(8) = 1 implies the printing out of time domain solution for variables of interest.

NOP(8) = 2,3 will store the data on disk for later "standard" plots on the scope, plotter.

The value of NOP(8) can be reset by using the plotting options described in section A.11.

NOP(9) permits the user to make changes to the problem specification or any of the program options through a terminal and then have his problem rerun. Some changes can be made with considerably less recalculation than others as can be seen from the flow diagram in Figure 2-1.

NOP(9) = 0 will cause an exit.

NOP(9) = 1 will cause the program to re-evaluate the time domain solution using the same $\underline{\phi}(T)$ and $\underline{H}(T)$ coefficients but with different $\underline{u}$.

NOP(9) = 2 will start with the matrix definition of the problem (whether generated by the program or entered by the user) and recalculate the time domain solution.

NOP(9) = 3 essentially allows the user to re-enter the program at the point where all of the original problem specifications (data cards) and subsequent changes are complete. The user can change and/or add to these previous specifications and rerun the solution.

Examples of the messages which assist the user in making changes using these options are given in Table A-3. These messages are generated at the completion of the previous solution in accordance with the current value of NOP(9).

A.10.2 Specifying Non-Standard Values for Option Vector

If the user wishes to change any special entry from its standard value he may do so either in the original data cards or, when requested, enter any desired changes via the keyboard.

The standard format for all program option changes is a "0,0" code followed by an "option ID" code given in Table A-4 and then the new values of the options.

Example: 0,0,1,8,2,3,1 **

where	0,0	implies a program option change
	1	implies a change to NOP vector
	8	identifies NOP(8)

TABLE A-3

Messages Assisting the User in Making Changes

```
Changing Disturbance Parameter(s): ............ NOP(9) = 1

ENTER 999,DIST.NO+100,DIST.CODE NO.,1ST DIST.VALUE,2ND DIST.VALUE**

999 101 1 2.0 0.0 **

IF A CHANGE OF ANY OF THE FOLLOWING IS DESIRED ENTER
  ID NO.S AND NEW VALUES **
  999 ** OTHERWISE OR AS LAST ENTRY
  1 WAY OF HANDLING THE OUTPUT...NOP(8)
  2 INITIAL TIME VALUE...............TO
  3 NO. OF TIME INTERVALS..........NPTS
  4 INITIAL STATE VECTOR...........X(0)
  5 OUTPUT VARIABLE VECTOR...........NY
  6 FURTHER CHANGES..............NOP(9)

5 3 **

999 **


Changing Program Control Entries (Options,etc.):NOP(9) = 2

ENTER ID NO.S AND NEW VALUES **
      999 ** AS LAST ENTRY

  1 OPTION(i)NO.,OPTION(i) VALUE.....**
  2 TO,DELT,DELTH....................**
  3 NUMBER OF POINTS.................**
  4 OUTPUT VARIABLE VECTOR...........**
  5 PRIORITY NUMBER FOR OUTPUT.......**

1 3 1 **
999 **


Changing Original Problem Specifications: ......NOP(9) = 3

ENTER T.F. TYPE OR PARAMETER **
ENTER NOP(9) VALUE TO STOP OR MAKE A DIFFERENT CHANGE

502 8 1 **
 502 8 1 **

999 0 **
 999 0 **
```

TABLE A-4

Program Control Entries

Code	Option ID	Description	reference
0,0,	1	vector of options	a)
0,0,	2	CRIT,TO,DELT,DELTH	b)
0,0,	3	NPTS	c)
0,0,	4	X(0)	d)
0,0,	5	NY	e)
0,0,	6	NUPOP	f)
0,0,	7	NJOBS	g)
0,	--	job number,NPRTY,LUNI, LUNO	h)
0,	--	LUNI,LUNO	i)

2 is the new value of NOP(8)

3 identifies NOP(3)

1 is the new value of NOP(3).

As many option changes as desired can be entered. Each change simply replaces the previous specification.

A.10.3 Specification of Time Parameters DELT, NPTS and DELTH

In order to determine the time domain response, $\underline{X}(t)$, at discrete intervals of time, DELT, the program generates the state difference equation

$$\underline{X}[(n+1)*DELT] = \underline{\phi}(DELT)*\underline{X}[n*DELT] + \underline{H}(DELT)*\underline{U}(n*DELT) \quad (A-6)$$

Starting with the specified initial conditions the program uses the above equation to generate the discrete series of values of the time domain solution

$$\underline{X}(0), \underline{X}(DELT), \underline{X}(2*DELT) \ldots \underline{X}(NPTS*DELT)$$

where NPTS is the number of points specified by the user. Thus DELT determines the interval for generating values of the time domain response and NPTS determines the number of discrete values generated.

If the forcing functions $\underline{U}$ are constant over each interval DELT (which usually means there can be no time delays in the problem) then the method of solution is exact and the discrete values generated by equation (A-6) should agree exactly with the continuous solution of the

original problem regardless of the values specified for DELT and NPTS. In this case DELT and NPTS are simply selected to give an adequate representation of the time domain solution over the total time of interest.

In a more typical case $\underline{U}$ are not exactly constant over any discrete interval and hence a relatively small interval DELT is assumed so that $\underline{U}$ from t = (n*DELT) to t = [(n+1)*DELT] can be adequately represented by a constant.

In other cases the interval DELT is fixed by the physical process: for example, a control device introducing step changes into the physical process at intervals of DELT.

Although the method is exact the numerical implementation of the method may generate errors due to round-off errors, etc. For example, the coefficient $\underline{H}$(DELT) in equation (A-6) is found by a numerical integration of $\underline{\phi}$(t)*$\underline{B}$ over the interval zero to DELT. This program uses the trapezoidal method of integration so the accuracy depends on the size of the integration step. The size of the integration step is DELTH and may be specified by the user. Experience to date indicates that DELTH = DELT/10 is usually sufficient. It is important to note that DELTH is an "internal" variable and does not affect the number or spacing of points in the time domain solution.

A.10.4 More Program Control Entries

With reference to Table A-4 "standard" values for program control entries b) to i) are now given.

b) CRIT = 3. It means that 10**(-CRIT) = 1% is taken as absolute criterion to simplify the eigenvalue problem.

TO = 0: initial time value for calculating the time domain response. Note that because of the recursive relation used for the time domain solution an other than zero initial time would require the user to supply the state variable vector at time = TO.

DELT = 0.1 this is the standard time interval.

DELTH = .01 this is the standard time subinterval to calculate $\underline{H}(T)$ equation (A-6).

c) NPTS = 10 total number of time intervals (or 11 points) Note that if NPTS = 0 then number of points $= \frac{1}{DELT} + 1)$.

d) $\underline{X}(0) = \underline{0}$ initial state vector is a null vector.

e) $\underline{NY} = \underline{Y}$ all output variables are requested in the time domain solution.

f) NUPOP = 0 no utility program is desired (see Table A-6).

g) NJOBS = 1 only one job is executed.

h) JOB NUMBER = 1 standard job number is 1.

NPRTY = 3 priority number for printing output is 3 (see Table A-5).

LUNI = 5 logical unit number for input device is 5 (card reader).

LUNO = 6 logical unit number for output device is 6 (1443 printer).

i) LUNI, LUNO as in h).

If the user wishes to change any program control entry from its standard value, he may do so by following the same procedure described for the options (section A.10.2), i.e., using the "0,0" code followed by the option ID given in Talbe A-4 and the new value(s). For example the user may change the logical unit numbers for I/0 device selecting an 1816 typewriter. In such a case the last card in the reader hopper would show as

i) 0,3,4,** or

h) 0,1,3,3,4,**

After this card has been read all I/0 are forced on the typewriter with LUN 3,4 and first among them the last entry 999 0 **

The advantage of doing so is to avoid messages on the printer where documentation of the problem solution is being printed. Another much more important advantage is that with terminal I/0 the user after looking at his first solution may make changes and have the solution rerun. Some of the more frequent changes in program parameter are collected in Table A-7.

TABLE A-5

Priority Number for Output

NPRTY = 3 implies a first level of documentation of a problem consisting of:

3.1 title if any

3.2 input data cards

3.3 all input data whether supplied by the user or by the program

3.4 the coefficient matrices of the state equation (A-1) and related vectors

3.5 the coefficient matrices of the state difference equation (A-2)

3.6 the time domain response of variables of interest

NPRTY = 5 implies a second level of documentation. The following output is added to the previous one.

5.1 the coefficients of the characteristic equation

5.2 the eigenvalues of coefficient matrix $\underset{\sim}{A}$

5.3 the shifted eigenvalues and the smallest one

5.4 the eigenvalues ordered in distinct, repeated and complex ones

5.5 the matrix $\underset{\sim}{CC}$ used to find the coefficients α_i according to the Cayley-Hamilton technique

5.6 the fundamental matrix $\underset{\sim}{\Phi}(t)$ at each time subinterval (DELTH)

NPRTY = 10 adds to the previous output all results required for the debugging stage. If necessary, further ouputs may be added under such option.

Examples of the printout produced by specifying options NPRTY = 3 and NPRTY = 5 are included in Appendix C.

TABLE A-6

Utility Program Options

NUPOP = 0	no utility program desired
NUPOP = 1 → 9	vacant
NUPOP = 10	coefficient matrices of the state difference equation (A-2) $\underline{\Phi}(T)$ and $\underline{H}(T)$ are punched out in format 5E15.6. Each matrix is preceded by a title. Enough blank cards must be stacked in the reader hopper after the last data card.

TABLE A-7

Input Cards for Frequently User-Specified Changes in Program Parameters

Change	Input Cards
1816 typewriter terminal I/O.	0,3,4,**
plot output variables on the scope	0,0,1,8,2,**
specify which element of Y (i,j,k,..) is to be plotted	0,0,5,i,j,k,..,**
permit the user to enter changes and rerun the problem	0,0,1,9,3,**

Note: The 0,3,4,** card is usually the last one in the user's input deck since all further input/output is done via the 1816 terminal.

A.11 Plotting

There are eight basic options which define the type of figure produced by the plotting subroutines. Each of these options is initially set by the program to the "standard" choice shown in Table A-8. The user is offered the choice of using these standard values or to changing one, or any number of the options. If the user has specified that input/output is to be the 1816 typewriter and that NOP(8) = 2 or NOP(8) = 3 at the beginning of the last problem solution then all further decisions about plotting are made as a result of questions generated by the program and answers entered by the user through the keyboard.

A plot using the standard options is shown in Figure A-2. What follows is an example of the messages produced by the program (upper case typing) and the responses by the user (underlined messages near the left margin). Additional single spaced explanatory messages have been added between program messages.

Note that these options can be used for example, to plot a result on the scope for visual examination and then to specify that the same results be plotted on the digital plotter. (No recalculation of the solution is involved, just data retreival from the disk file).

Table A-8

Standard Plotting Options

Option Number	Description
1	YMIN and YMAX values for the ordinate of the graph as found by searching the file of data to be plotted.
2	the plot is produced on a new area rather than over top of a previous graph.
3	the figure is produced on either the scope or the digital plotter depending on the specification of NOP(8).
4	the Y data is plotted as a continuous line produced by linear interpolation between the data points (other options are to use a + or x for each data point).
5	standard figures are plotted two per 8.1/2" x 11" page. The first graph is produced on the top half of the page.
6	the standard label for the X axis is "T/TMAX" where T = time and TMAX = NPTS*DELT.
7	the standard label for the Y axis in "Y(n)" where n is an integer constant identifying the particular output variable plotted.
8	the standard title is "Time Domain Response".

Note: a "standard" graph is shown in Figure A-2.

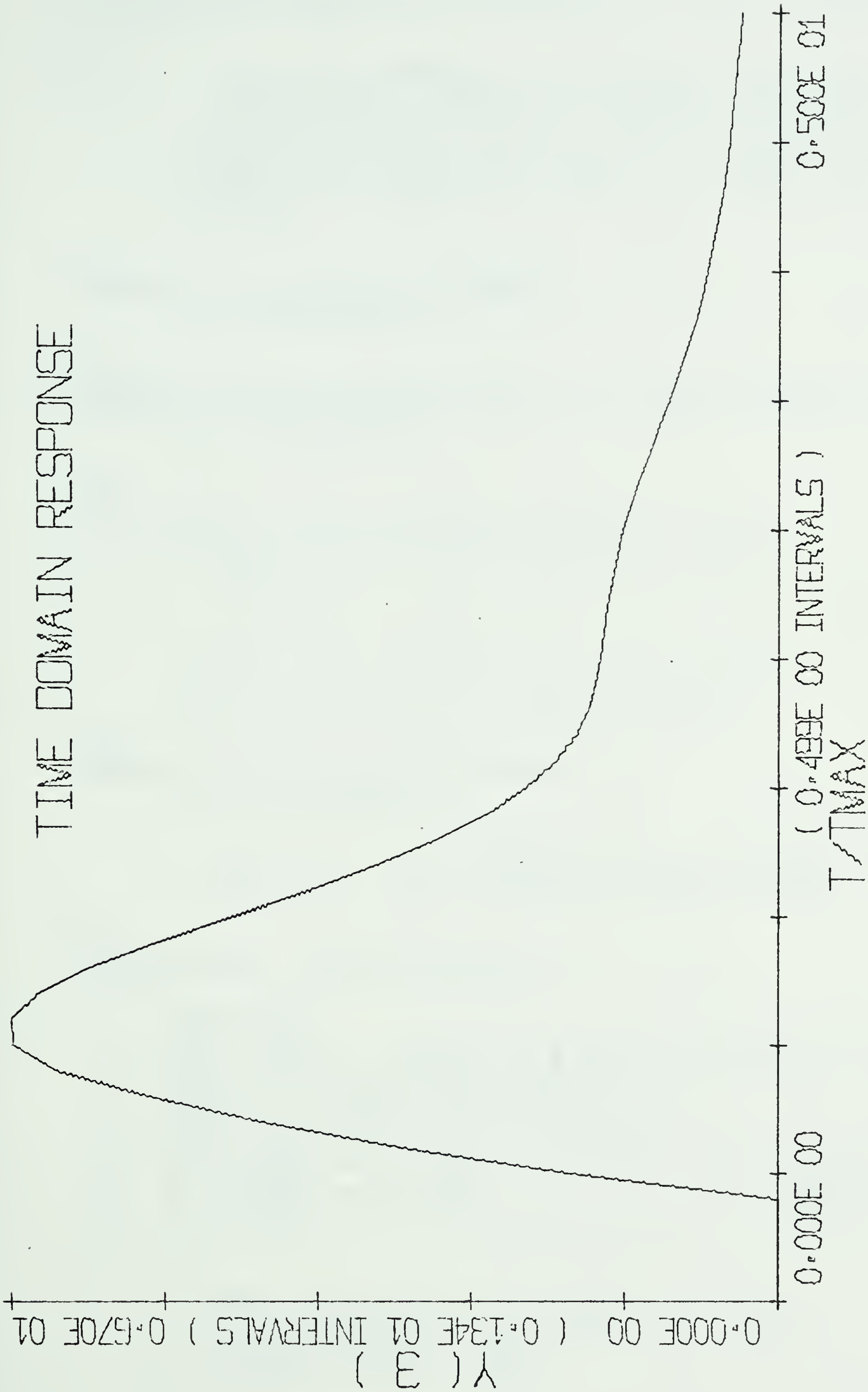


FIGURE A-2 Example of Standard Plot

Example of Input/Output to Produce a User Selected Plot

The first message is printed after the program has solved the problem specified by the user and stored the time domain data in a disk file. Note that the user must have specified LUNI = 3, LUNO = 4, NOP(8) = 2 or 3.

```
ENTER 1 ** IF PLOT IS DESIRED
      0 ** OTHERWISE

1 **
ENTER 1 ** FOR PLOTTING ALL YS IN THE STANDARD FORM
      0 ** OTHERWISE

0 **
ENTER ID NO. FOR DESIRED VARIABLE Y

         ID          Y

         1           1
         2           2
         3           3

3 **
ENTER 1 ** FOR STANDARD GRAPH
      0 ** OTHERWISE
```

If a 1 ** was entered here the program would plot Y(3) in a standard figure format.

```
0 **
ENTER ID NO. AND NEW VALUES **

 1   YMIN,YMAS
 2   PLOT STATUS..1-NEW,2=OVERTOP,3=OVERBOTTOM..**
 3   PLOT TYPE..-VE=LINE,0=+,1=X ETC............**
 4   DEVICE..1=SCOPE,2=PLOTTER..................**
 5   PLOT POSITION..1=TOP,2=BOTTOM..............**
 6   **..LABEL ON X AXIS.............*..(20A1)..*
 7   **..LABEL ON Y AXIS.............*..(20A1)..*
 8   **..TITLE.......................*..(40A1)..*
```

When entering a new label or title, blanks should be used to fill the allowed input field.

```
LAST ENTRY IS 999 **
```

1 0.0 20.0 **
2 2 **
3 5 **
999 **

Program plots the requested figure at this point and then returns to see if the user wishes any additional figures plotted.

ENTER 1 ** IF PLOT IS DESIRED
0 ** OTHERWISE

0 **

ENTER NOP(9) VALUE FOR FURTHER CHANGES (format Ib**)
0 ** OTHERWISE

0 **

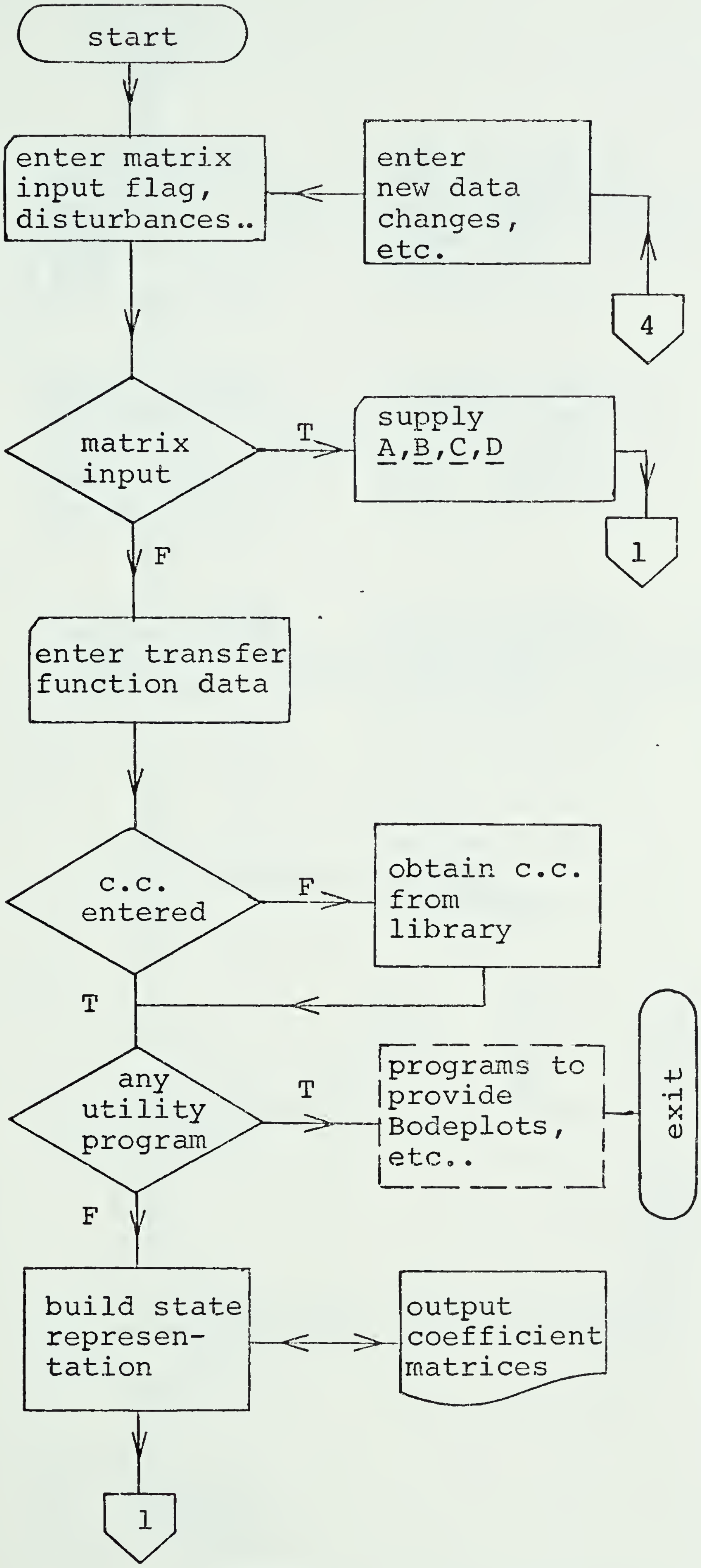
The program terminates at this point. NOTE that the specification of NOP(9) = 3 by typing in the following data:

3,**b would essentially return the user to the beginning of the program where he could make any changes he wished and re-execute the problem.

APPENDIX B

Program Flow Diagrams

comments:

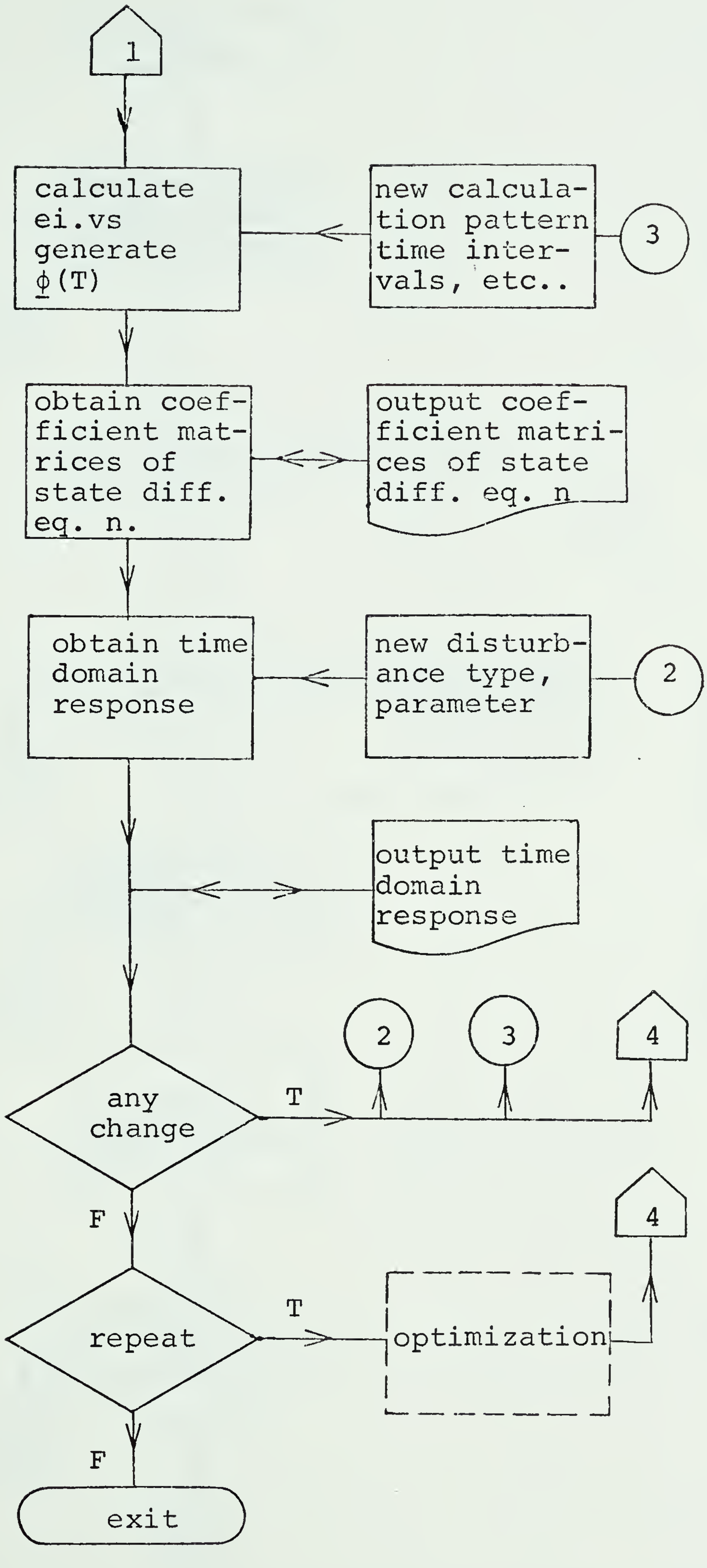


c.c. = controller constants

--- = program not implemented

...continue

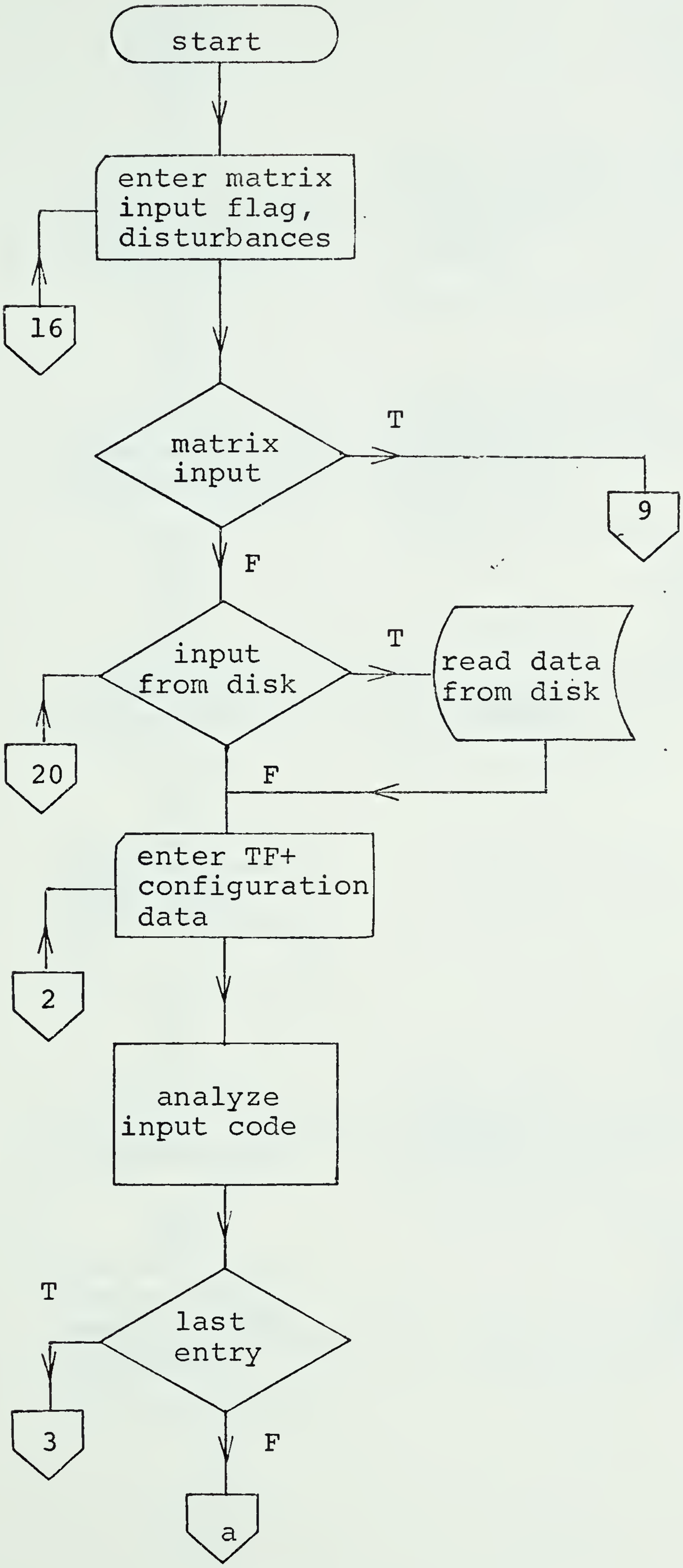
FIGURE B-1 General Flow Diagram



comments:

ei.vs = eigenvalues of matrix A

$\underline{\phi}(T)$ = fundamental matrix



comments

Link LAST1 starts

these entries are common to the two types of input

test:MFLAG.GT.0

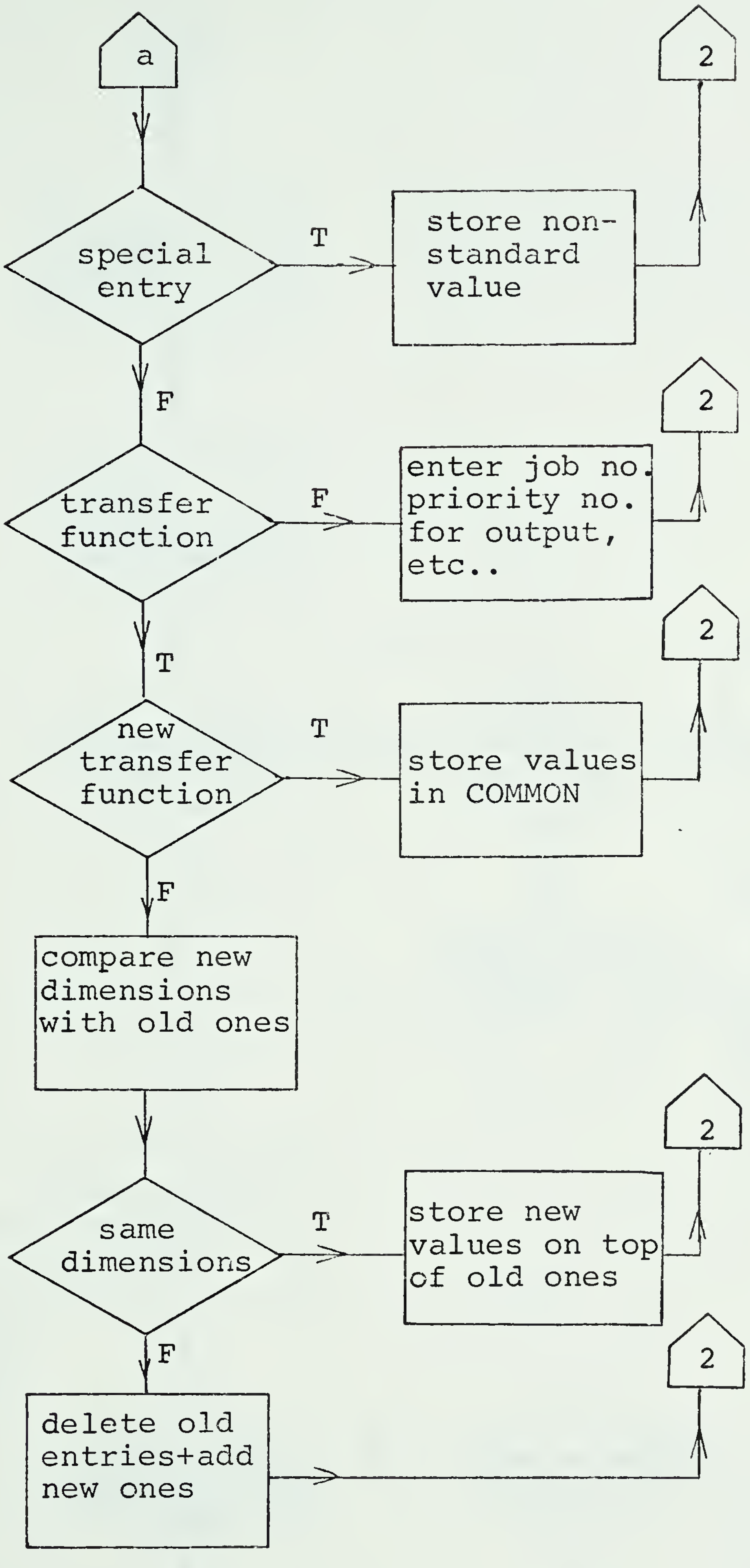
Test:IFDFG.GT.0
Recalls data and problem specifications entered during previous runs.

TF=transfer function permits entry of additions, changes or new problems

test:lst value = 999

...continue

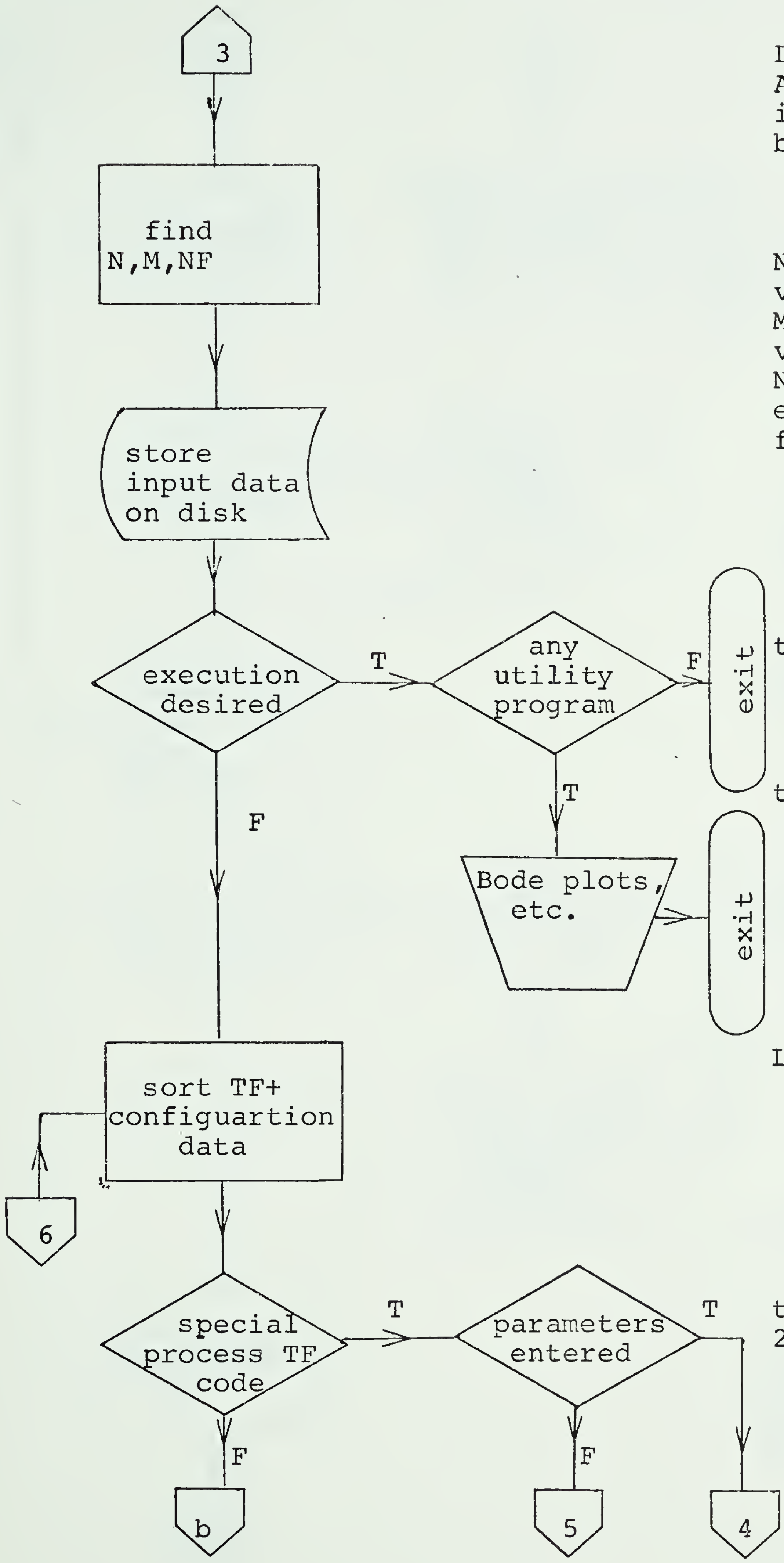
FIGURE B-2 Detailed Flow Diagram



test:1st entry= 0,0,...

test:0.LT.1st entry.LT.999

...continue



Link LAST2 starts.
At this point all
information has
been entered

N:number of state
variables
M:number of output
variables
NF:number of
external forcing
functions

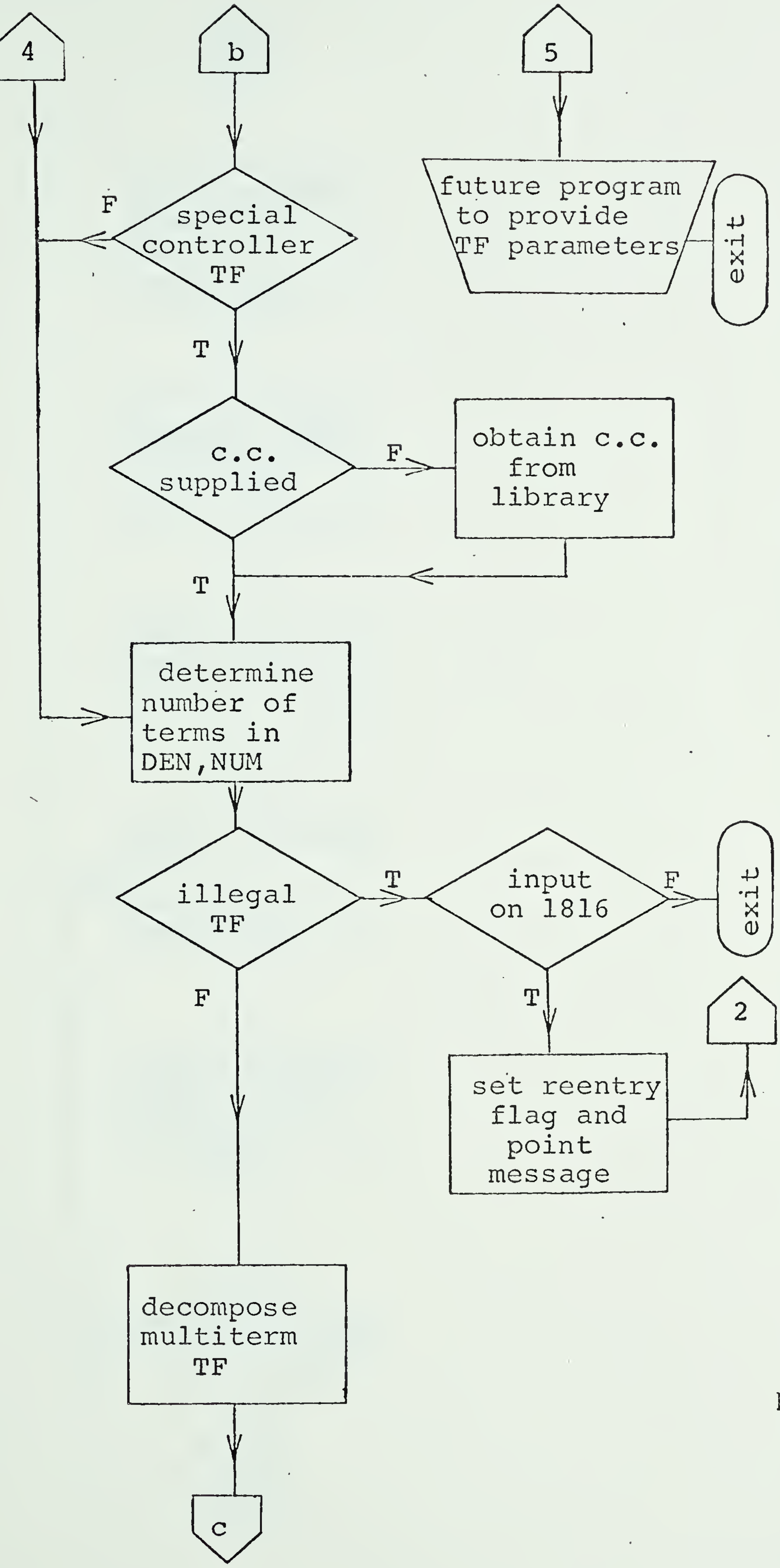
test:NEXFG=0

test:NUPOP=1,..9

Link LAST3 starts

test:1st entry=
200+ID,300+ID

...continue



test:1st entry=(400,500,600)+ID

c.c.:controller constants

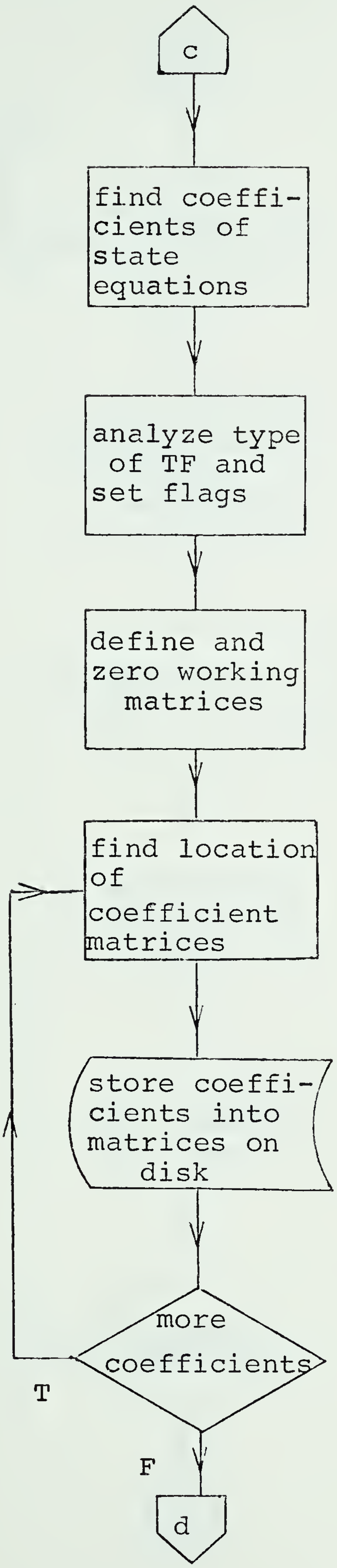
DEN=denominator
NUM=numerator

reentry flag:
IFLG1=99

Link LAST4 starts.

TF are manipulated into the ratio of two polynomials in "s"

...continue



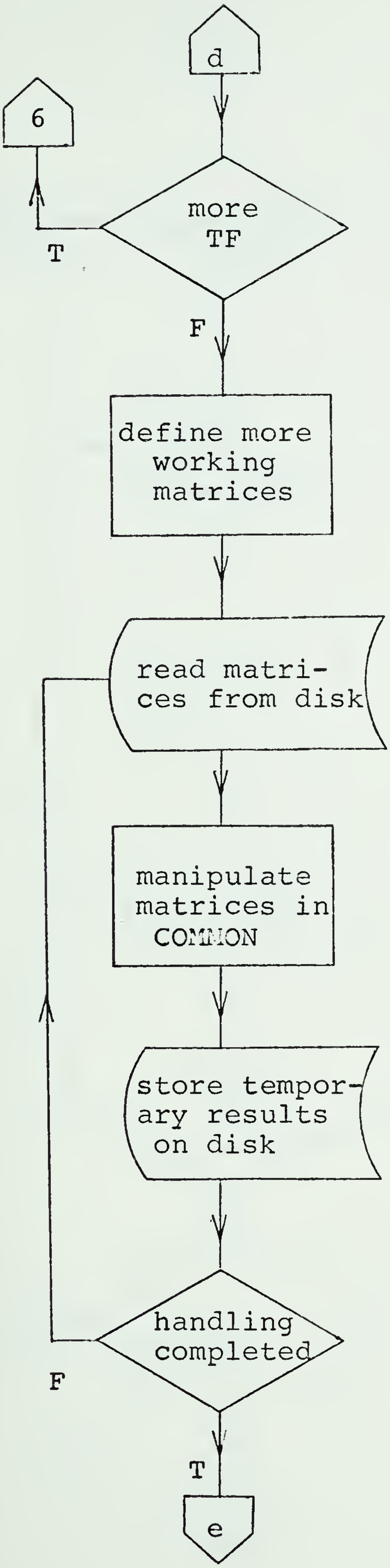
the TF is transformed into a system of first order differential equations. State variables are introduced at this point.

the program analyzes TF's, steps accumulators and sets flags for special cases
Link LAST5 starts

working matrices are:
A1, B1, C1, D1, E1, D2, E2, F,G, H,AL,AM,AL1

each is identified as a disk file

... continue

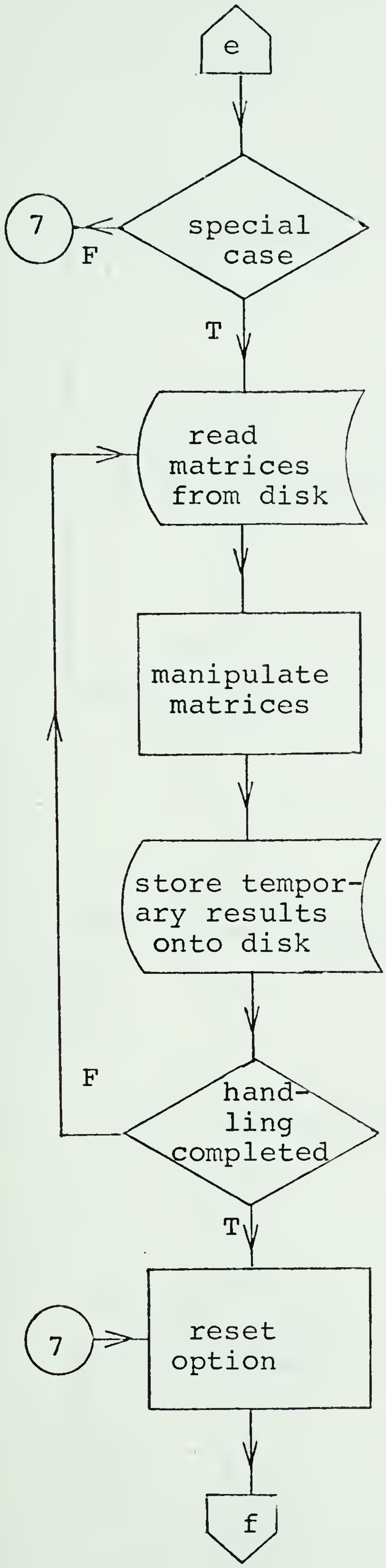


Link LAST6 starts

these extra matrices are used to manipulate the previous ones into standard format

$$\dot{\underline{X}} = \underline{\underline{A}}\,\underline{X} + \underline{\underline{B}}\,\underline{U}$$
$$\underline{Y} = \underline{\underline{C}}\,\underline{X} + \underline{\underline{D}}\,\underline{V}$$

...continue



Link LAST7 starts.

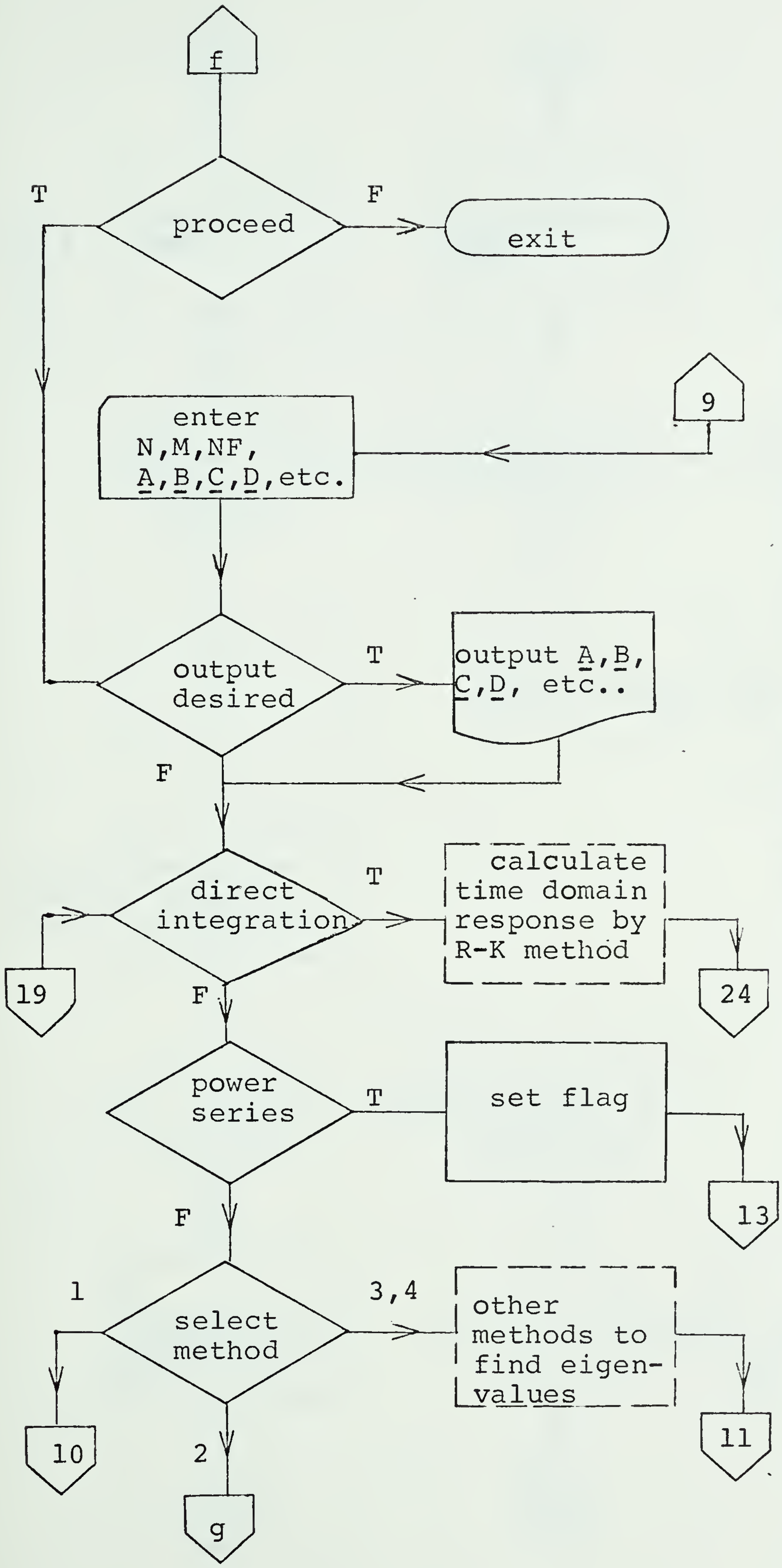
test:NFLG1,NFLG2
NFLG3.GT.0

manipulation for these special cases is quite different

Special cases include:
- isolated time delays
- isolated derivative terms

NOP(7) is reset according to the type of problem i.e.
NOP(7)=1 if $\underline{V}=\underline{O}$
NOP(7)=0 if $\underline{V}=\underline{C}=\underline{O}$

...continue



Formulation of state variable representation is complete at this point. Exit if $\underline{u}$=o

Link WAF1 starts.

It continues the input of data in the case of "matrix input"

Link WAF2 starts. It provides a record of input data

Link WAF3 starts.

linkage is provided to direct integration of state equation by Runga-Kutta or other methods test: NOP(2).GT.0

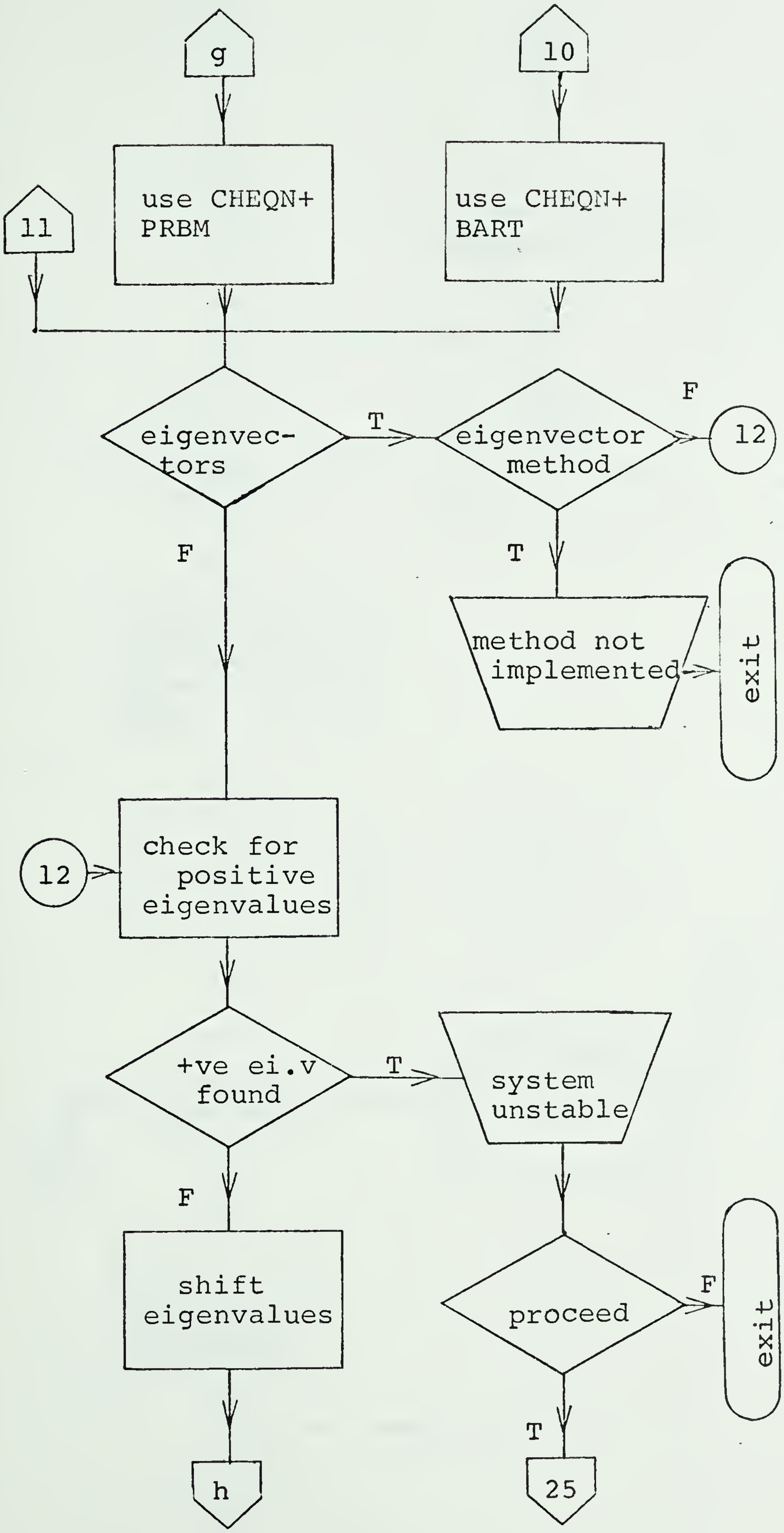
---program not implemented

find eigenvalues of coefficient matrix $\underline{A}$

test:NOP(4)=1,2, 3,4

power series test is:NOP(3)=1

...continue



there are two different subroutines to find the eigenvalues

if the matrix of the eigenvectors is found as well the following similarity transformation may be used to calculate the fundamental matrix:

$\underline{A}=\underline{M}\underline{\Lambda}\underline{M}^{-1}$

$\underline{\Lambda}$=diagonal matrix of distinct eigenvalues

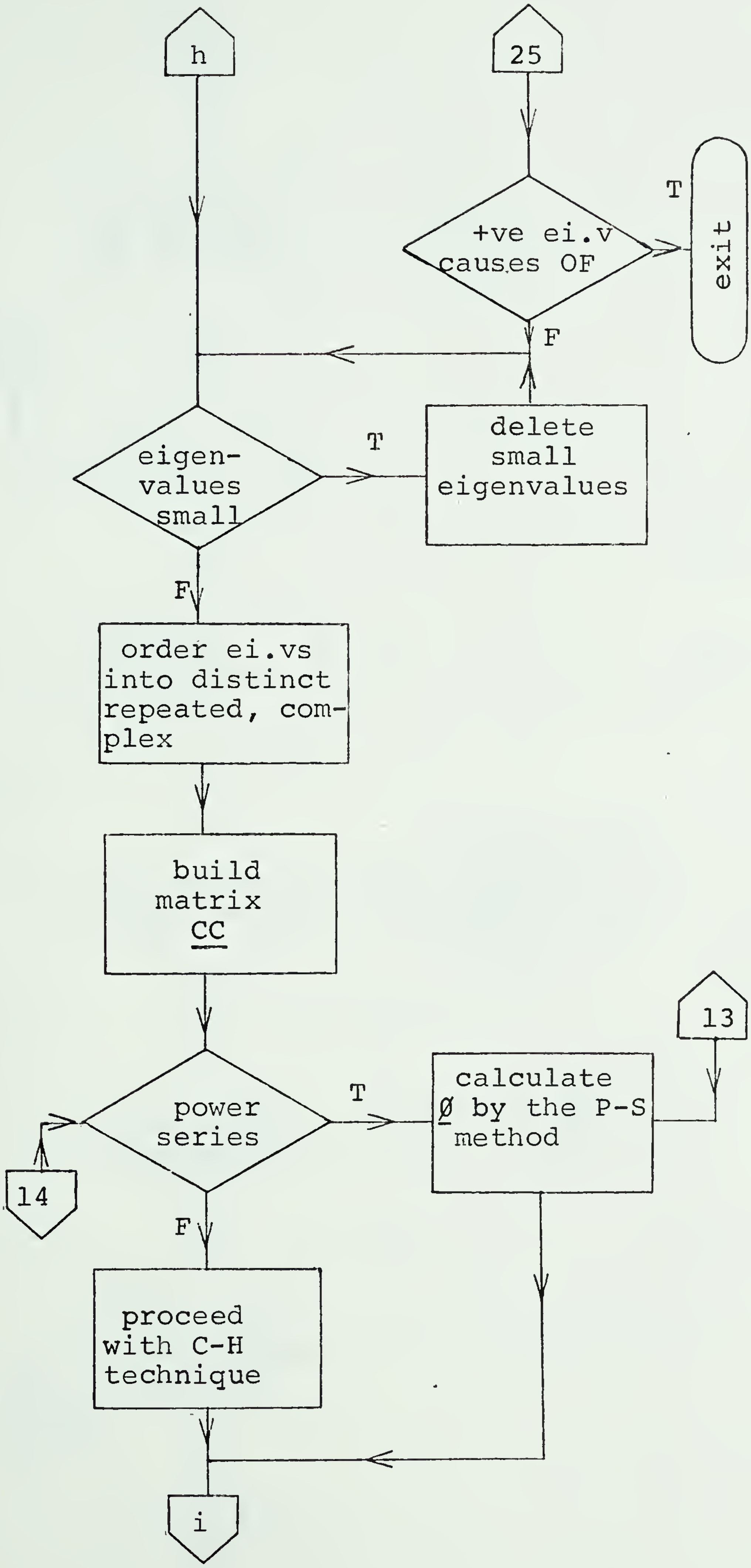
M=matrix of eigenvectors

Test: NOP(5).GT.0

Link WAF4 starts.

test:NOP(6).GT.0

...continue



OF=overflow.Test is done to check that positive eigenvalues do not cause over= flow(LT.EXP(80))

this is a test to compare the effect of eigenvalues on the final response

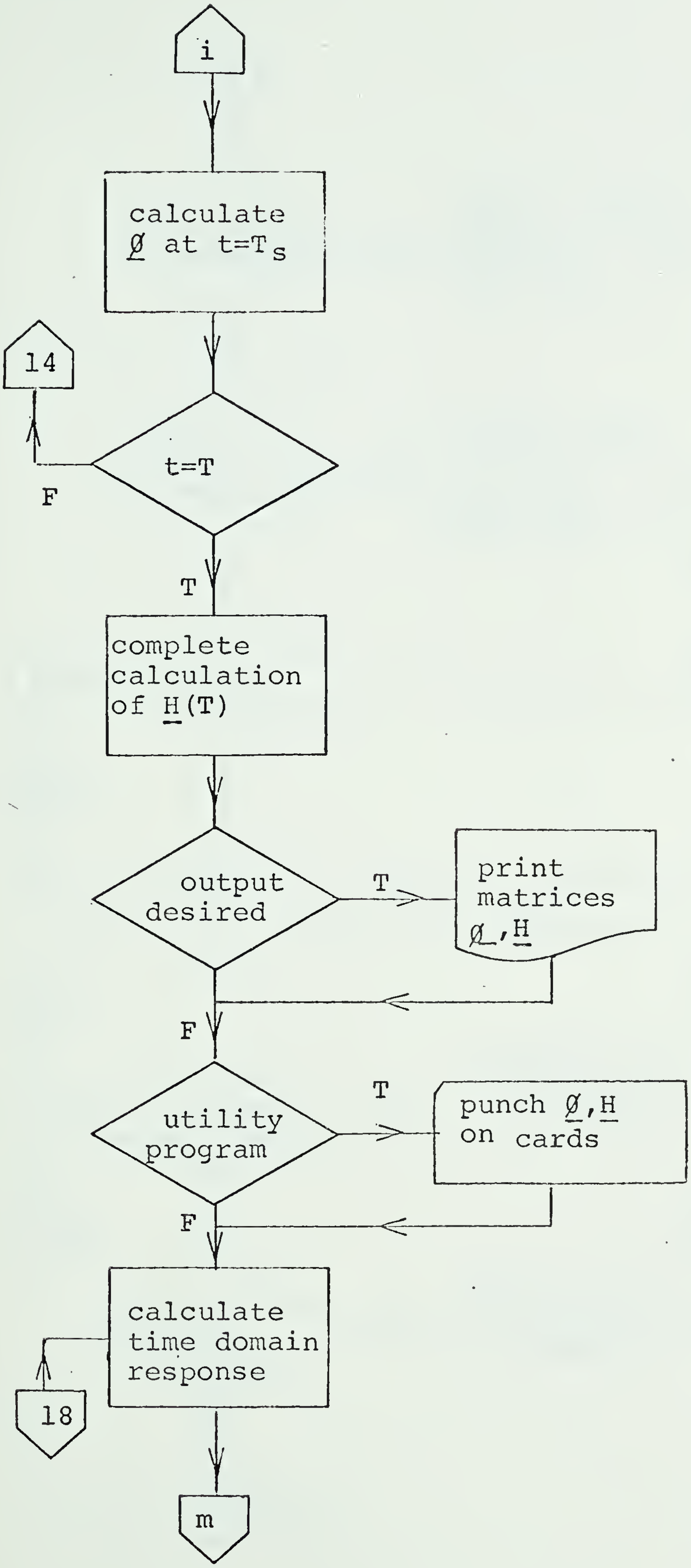
matrix CC is used to find the fun- damental matrix according to the Cayley-Hamilton technique

Link WAF5 starts.

test:MPSFG=99
P-S: power series

C-H: Cayley- Hamilton

...continue



T_s = time sub-interval

$\underline{H}(T)$=integral between 0 and T of:

$\underline{\emptyset}(t)\underline{B}dt$

is calculated according to the trapezoidal rule

Link WAF6 starts

$\underline{\emptyset}$,$\underline{H}$ are coefficient matrices of the state difference equations:

$$\underline{X}_{i+1} = \underline{\emptyset}(T)\underline{X}_i + \underline{H}(T)\underline{U}_i$$

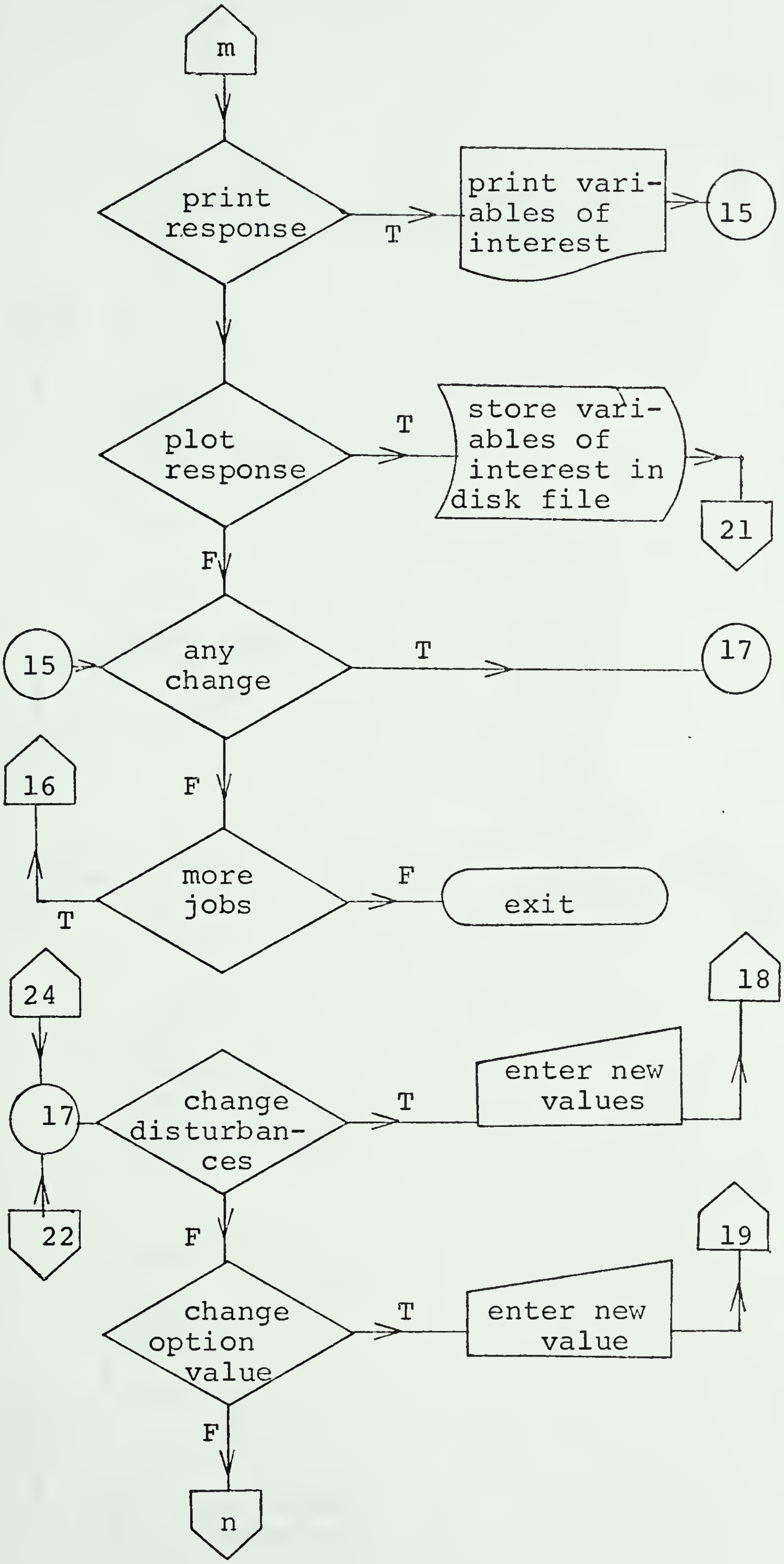
$$\underline{Y}_i = \underline{C}\underline{X}_i + \underline{D}\underline{V}_i$$

test for output: NPRTY.GE.3

test:NUPOP=10

Link WAF7 starts

...continue



test:NOP(8)=1 or NPRTY.GE.3

test:NOP(8)=2,3

test:NOP(9).GT.0

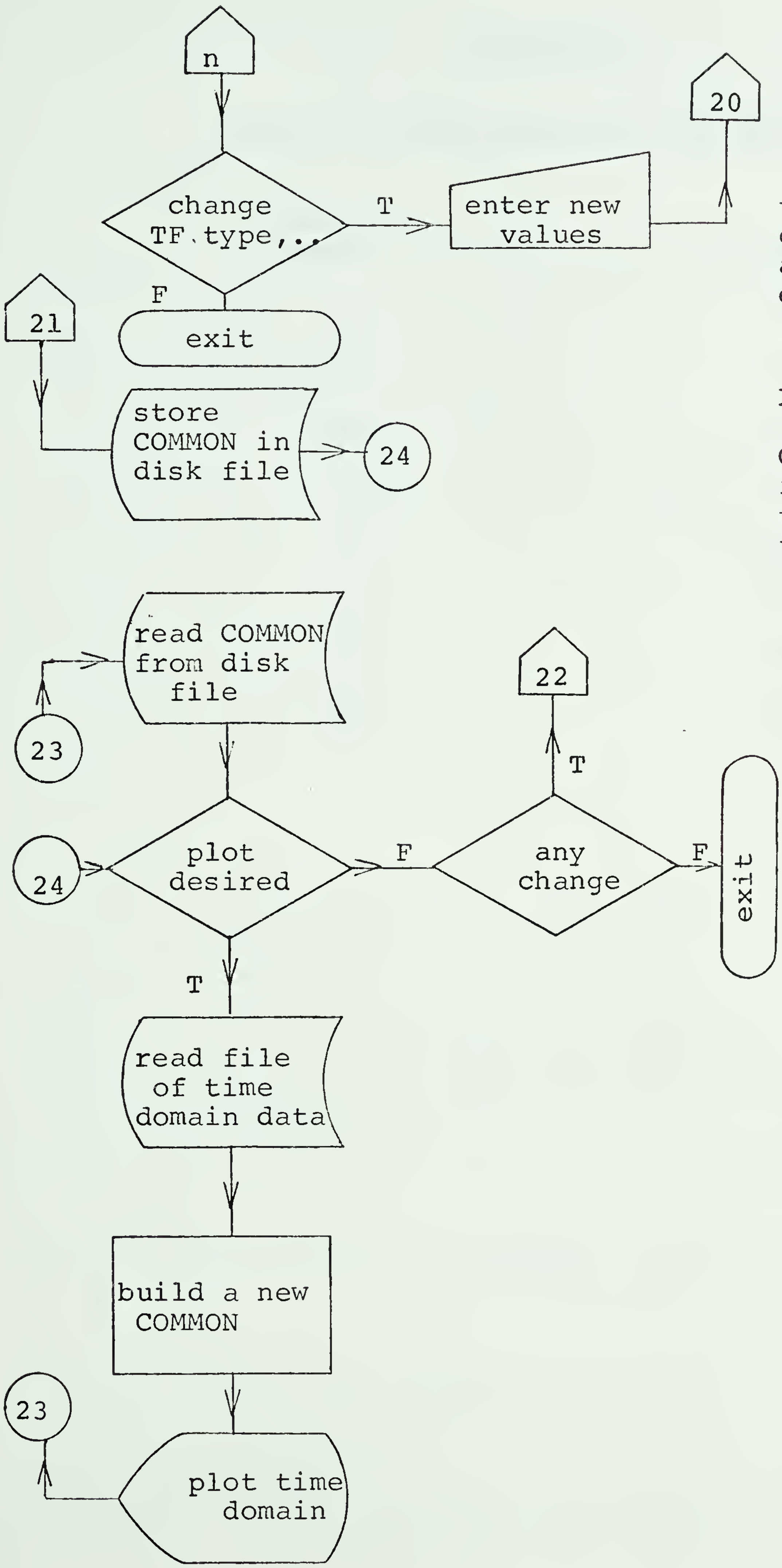
test:NJOBS.GT.1

Link WAF8 starts

test:NOP(9)=1 change disturbance type and/or parameter

test:NOP(9)=2 change option value time interval, etc.

...continue



test:NOP(9)=3
change TF type
and/or parameter
etc...

Link WAF10 starts

COMMON is saved
in a disk file
the first time
through

APPENDIX C

Complete Documentation of the Examples

```
                                          * JULY 69   *

             * THIS IS A TYPICAL MATRIX INPUT PROBLEM *
             * FIVE EQUATIONS EVAPORATOR MODEL EXAMPLE *
             * THIS RUN USES CAYLEY-HAMILTON TECHNIQUE*

 * INPUT DATA CARDS *

1                          **
* THIS IS TO CONTINUE FOR POSITIVE EIGENVALUE*
0 0 1 6 1                  **
* THIS DEFINES THE TYPE OF PROBLEM *
0 0 1 7 1                  **
* THIS SUPPLIES CRIT,TO,DELT,DELTH *
0 0 2 3.0 0.0 0.0167 0.00167       **
* THIS SUPPLIES THE INITIAL STATE VECTOR X(0) *
0 0 4  37.5 4.34 152.3 26.9 7.51   **
* THIS GIVES THE NUMBER OF TIME INTERVALS *
0 0 3 10                   **
* STEADY STATE VALUES ARE ENTERED AS STEP DISTURBANCES *
999 101 1 1.0 0.00         **
999 102 1 1.69 0.00        **
999 103 1 0.98 0.00        **
999 104 1 2.43 0.00        **
999 105 1 3.02 0.00        **
999 106 1 55.3 0.00        **
999 0                      **
5 3 6                      **
0                          **
* THIS IS MATRIX A *
0.0 0.0 0.0 0.0 0.0 -0.002 -0.043 -0.014 -0.003 0.039
-0.087 0.087 -0.571 -0.125 0.125 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 -0.0001 -0.036 **
* THIS IS MATRIX B *
 0.00 0.00 0.169 0.00 0.00 -0.045 0.00 0.00 **
0.061 -.024 0.00 0.00 0.00 -.036 0.00 **
 0.065 -0.02 -0.04 0.00 0.00 0.00 0.045 0.00 0.00 **
 0.00 0.00 0.00 0.025 0.00 0.00 **
* THIS IS MATRIX C *
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0
0.0 1.0 **

END OF INPUT DATA FOR JOB NUMBER    1


                         MATRIX INPUT


INPUT MATRIX FLAG.........................        1
JOB NUMBER................................        1
```

```
PRIORITY NUMBER FOR PRINTING OUTPUT.....            3
LOGICAL UNIT NUMBER FOR INPUT DEVICE....            5
LOGICAL UNIT NUMBER FOR OUTPUT DEVICE...            6

OPTION.. 1..............................            1
OPTION.. 2..............................            0
OPTION.. 3..............................            0
OPTION.. 4..............................            2
OPTION.. 5..............................            0
OPTION.. 6..............................            1
OPTION.. 7..............................            1
OPTION.. 8..............................            1
OPTION.. 9..............................            0
OPTION..10..............................            0

ABSOLUTE CRITERION VALUE (PER CENT).....       0.1000
INITIAL TIME VALUE......................       0.0000
TIME INTERVAL...........................       0.0167
TIME SUBINTERVAL........................       0.0016
NUMBER OF TIME INTERVALS................           10

INITIAL STATE VARIABLE.. 1..............      37.5000
INITIAL STATE VARIABLE.. 2..............       4.3400
INITIAL STATE VARIABLE.. 3..............     152.3000
INITIAL STATE VARIABLE.. 4..............      26.9000
INITIAL STATE VARIABLE.. 5..............       7.5100

DESIRED OUTPUT/STATE VARIABLE.. 1.......            1
DESIRED OUTPUT/STATE VARIABLE.. 2.......            2
DESIRED OUTPUT/STATE VARIABLE.. 3.......            3

UTILITY PROGRAM NUMBER..................            0
NUMBER OF JOBS..........................            1

DIST.. 1  CODE.. 1  1ST PAR..    1.0000  2ND PAR..      0.0000
DIST.. 2  CODE.. 1  1ST PAR..    1.6900  2ND PAR..      0.0000
DIST.. 3  CODE.. 1  1ST PAR..    0.9800  2ND PAR..      0.0000
DIST.. 4  CODE.. 1  1ST PAR..    2.4299  2ND PAR..      0.0000
DIST.. 5  CODE.. 1  1ST PAR..    3.0200  2ND PAR..      0.0000
DIST.. 6  CODE.. 1  1ST PAR..   55.3000  2ND PAR..      0.0000


         COEFFICIENT MATRICES OF THE STATE EQUATION
                      AND RELATED DATA


NUMBER OF STATE VARIABLES...............            5
NUMBER OF OUTPUT VARIABLES..............            3
NUMBER OF EXTERNAL FORCING FUNCTIONS....            6
NUMBER OF TIME DELAYS...................            0

NUMBER OF EXTERNAL FORCING FUNCTIONS IN U(T)        6
```

```
VARIABLE IN U(T).... 1...DELAY VALUE....    0.0000
VAPIABLE IN U(T).... 2...DFLAY VALUE....    0.0000
VARIABLE IN U(T).... 3...DELAY VALUE....    0.0000
VARIABLE IN U(T).... 4...DELAY VALUE....    0.0000
VARIABLE IN U(T).... 5...DELAY VALUE....    0.0000
VARIABLE IN U(T).... 6...DELAY VALUE....    0.0000
```

MATRIX A

0.0000	-0.0020	-0.0870	0.0000	0.0000
0.0000	-0.0430	0.0870	0.0000	0.0000
0.0000	-0.0140	-0.5709	0.0000	0.0000
0.0000	-0.0030	-0.1250	0.0000	-0.0001
0.0000	0.0390	0.1250	0.0000	-0.0360

MATRIX B

0.0000	-0.0450	0.0000	0.0650	0.0000	0.0000
0.0000	0.0000	0.0000	-0.0200	0.0450	0.0000
0.1689	0.0000	0.0000	-0.0400	0.0000	0.0250
0.0000	0.0609	-0.0360	0.0000	0.0000	0.0000
0.0000	-0.0240	0.0000	0.0000	0.0000	0.0000

MATRIX C

1.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	1.0000	0.0000
0.0000	0.0000	0.0000	0.0000	1.0000

POSITIVE ROOT FOUND - SYSTEM UNSTABLE
PROGRAM IS CONTINUED AS REQUESTED

COEFFICIENT MATRICES OF THE STATE
DIFFERENCE EQUATION

MATRIX FM(T)

0.9999E 00	-0.3321E-04	-0.1446E-02	0.0000E 00	0.0000E 00
0.0000E 00	0.9992E 00	0.1445E-02	0.0000E 00	0.0000E 00
0.0000E 00	-0.2326E-03	0.9905E 00	0.0000E 00	0.0000E 00
0.0000E 00	-0.4983E-04	-0.2077E-02	0.9999E 00	-0.1669E-05
0.0000E 00	0.6505E-03	0.2077E-02	0.0000E 00	0.9993E 00

MATRIX H(T)

-0.2043E-05	-0.7514E-03	0.0000E 00	0.1085E-02	-0.1250E-07	-0.3023E-06
0.2043E-05	0.0000E 00	0.0000E 00	-0.3343E-03	0.7512E-03	0.3022E-06
0.2808E-02	0.0000E 00	0.0000E 00	-0.6647E-03	-0.8754E-07	0.4155E-03
-0.2936E-05	0.1018E-02	-0.6011E-03	0.7033E-06	-0.1875E-07	-0.4343E-06
0.2936E-05	-0.4006E-03	0.0000E 00	-0.8036E-06	0.2445E-06	0.4343E-06

TIME DOMAIN RESPONSE

TIME	Y(1)	Y(2)	Y(3)
0.0000	37.5000	26.9000	7.5100
0.0167	37.2809	26.5844	7.8240
0.0334	37.0639	26.2718	8.1350
0.0500	36.8489	25.9620	8.4431
0.0668	36.6359	25.6552	8.7462
0.0834	36.4249	25.3513	9.0504
0.1001	36.2159	25.0501	9.3497
0.1168	36.0088	24.7518	9.6461
0.1336	35.8037	24.4563	9.9398
0.1502	35.6005	24.1635	10.2306
0.1669	35.3992	23.8734	10.5186

```
                                                    * JULY 69 *

                   * IN THIS EXAMPLE PI CONTROLLER CONSTANTS *
                   * FROM LIBRARY ARE COMPARED FOR A UNIT     *
                   * STEP CHANGE IN SET POINT                 *
                   * RATIO OF TIME DELAY TO TIME LAG IS 0.8   *
                   * PROCESS GAIN IS 10                       *

         * INPUT DATA CARDS *

* THIS IS THE MATRIX INPUT FLAG *
0                          **
* THIS IS TO OBTAIN A PLOT OF TIME DOMAIN DATA *
0 0 1 8 3                  **
* THIS IS THE NUMBER OF TIME INTERVALS =50 *
0 0 3 50                   **
* THIS IS TO PROCEED FOR POSITIVE EIGENVALUES*
0 0 1 6 1                  **
1 0 1.0 0.0                **
101 101 3 1.0 -1.0         **
203 10.0 0.4 0.5           **
103 2 1.0                  **
502 1 1                    **
102 1 1.0                  **
* THIS IS THE FORCING FUNCTION *
999 101 1 1.0 0.0          **
* THIS IS TO CHANGE THE LOGICAL UNIT NUMBERS *
0 3 4                      **


                 CONTROLLER CONSTANTS FROM LIBRARY

PROCESS GAIN.............................    10.0000
TIME DELAY...............................     0.4000
TIME LAG.................................     0.5000
CONTROLLER CODE..........................          5
CODED CRITERION TO SELECT CONSTANTS......          1
CODED DISTURBANCE TO SELECT CONSTANTS....          1

PROPORTIONAL CONSTANT....................     0.1125
INTEGRAL TIME............................     1.3332
DERIVATIVE TIME..........................     0.0000


                      BLOCK DIAGRAM INPUT


INPUT MATRIX FLAG........................          0
JOB NUMBER...............................          1
PRIORITY NUMBER FOR PRINTING OUTPUT......          3
LOGICAL UNIT NUMBER FOR INPUT DEVICE.....          3
```

```
LOGICAL UNIT NUMBER FOR OUTPUT DEVICE...              4

OPTION.. 1...............................             0
OPTION.. 2...............................             0
OPTION.. 3...............................             0
OPTION.. 4...............................             2
OPTION.. 5...............................             0
OPTION.. 6...............................             1
OPTION.. 7...............................             2
OPTION.. 8...............................             3
OPTION.. 9...............................             0
OPTION..10...............................             0

ABSOLUTE CRITERION VALUE (PER CENT).....         0.1000
INITIAL TIME VALUE......................         0.0000
TIME INTERVAL...........................         0.1000
TIME SUBINTERVAL........................         0.0100
NUMBER OF TIME INTERVALS................             50

INITIAL STATE VARIABLE.. 1..............         0.0000
INITIAL STATE VARIABLE.. 2..............         0.0000

DESIRED OUTPUT/STATE VARIABLE.. 1.......              1
DESIRED OUTPUT/STATE VARIABLE.. 2.......              2
DESIRED OUTPUT/STATE VARIABLE.. 3.......              3

UTILITY PROGRAM NUMBER..................              0
NUMBER OF JOBS..........................              1

DIST.. 1  CODE.. 1  1ST PAR..    1.0000  2ND PAR..    0.0000
```

COEFFICIENT MATRICES OF THE STATE EQUATION
AND RELATED DATA

```
NUMBER OF STATE VARIABLES...............              2
NUMBER OF OUTPUT VARIABLES..............              3
NUMBER OF EXTERNAL FORCING FUNCTIONS....              1
NUMBER OF TIME DELAYS...................              1

NUMBER OF EXTERNAL FORCING FUNCTIONS IN U(T)          1
DELAYED EXTERNAL FORCING FUNCTIONS IN...U(T)          1
DELAYED STATE VARIABLES IN..............U(T)          2

VARIABLE IN U(T).... 1...DELAY VALUE....         0.0000
VARIABLE IN U(T).... 1...DELAY VALUE....         0.4000
VARIABLE IN U(T).... 1...DELAY VALUE....         0.4000
VARIABLE IN U(T).... 2...DELAY VALUE....         0.4000

NUMBER OF EXTERNAL FORCING FUNCTIONS IN V(T)          1

VARIABLE IN V(T).... 1...DELAY VALUE....         0.0000
```

MATRIX A

-2.0000	0.0000
-0.0843	0.0000

MATRIX B

0.0000	2.2499	-2.2499	20.0000
0.0843	0.0000	0.0000	0.0000

MATRIX C

-1.0000	0.0000
-0.1124	1.0000
1.0000	0.0000

MATRIX D

1.0000
0.1124
0.0000

COEFFICIENT MATRICES OF THE STATE
DIFFERENCE EQUATION

MATRIX FM(T)

0.8187E 00	0.0000E 00
-0.7648E-02	0.1000E 01

MATRIX H(T)

0.0000E 00	0.2039E 00	-0.2039E 00	0.1812E 01
0.8438E-02	-0.8887E-03	0.8887E-03	-0.7900E-02

TIME DOMAIN RESPONSE

TIME	Y(1)	Y(2)	Y(3)
0.0000	1.0000	0.1124	0.0000
0.1000	1.0000	0.1209	0.0000
0.2000	1.0000	0.1293	0.0000
0.3000	1.0000	0.1378	0.0000
0.4000	1.0000	0.1462	0.0000
0.5000	0.7960	0.1308	0.2039
0.6000	0.6138	0.1162	0.3861
0.7000	0.4492	0.1022	0.5507
0.8000	0.2992	0.0884	0.7007
0.9000	0.1611	0.0748	0.8388
1.0000	0.0760	0.0662	0.9239
1.1000	0.0327	0.0618	0.9672
1.2000	0.0227	0.0609	0.9772
1.3000	0.0394	0.0631	0.9605
1.4000	0.0778	0.0679	0.9221
1.5000	0.1248	0.0740	0.8751
1.6000	0.1713	0.0805	0.8286
1.7000	0.2109	0.0866	0.7890
1.8000	0.2395	0.0917	0.7604
1.9000	0.2542	0.0955	0.7457
2.0000	0.2551	0.0977	0.7448
2.1000	0.2440	0.0986	0.7559
2.2000	0.2239	0.0983	0.7760
2.3000	0.1982	0.0972	0.8017
2.4000	0.1704	0.0956	0.8295
2.5000	0.1435	0.0939	0.8564
2.6000	0.1200	0.0924	0.8799
2.7000	0.1012	0.0912	0.8987
2.8000	0.0879	0.0905	0.9120
2.9000	0.0798	0.0903	0.9201
3.0000	0.0763	0.0905	0.9236
3.1000	0.0762	0.0912	0.9237
3.2000	0.0783	0.0921	0.9216
3.3000	0.0813	0.0931	0.9186
3.4000	0.0840	0.0941	0.9159
3.5000	0.0859	0.0950	0.9140
3.6000	0.0862	0.0958	0.9137
3.7000	0.0848	0.0963	0.9151
3.8000	0.0819	0.0967	0.9180
3.9000	0.0777	0.0969	0.9222
4.0000	0.0726	0.0970	0.9273
4.1000	0.0670	0.0969	0.9329
4.2000	0.0614	0.0968	0.9385
4.3000	0.0561	0.0967	0.9438
4.4000	0.0514	0.0967	0.9485
4.5000	0.0475	0.0966	0.9524
4.6000	0.0444	0.0967	0.9555
4.7000	0.0419	0.0968	0.9580
4.8000	0.0401	0.0969	0.9598
4.9000	0.0388	0.0971	0.9611
5.0000	0.0377	0.0973	0.9622

```
999 0 **
 999 0 **

 END OF INPUT DATA FOR JOB NUMBER     1

 INPUT OF A T.F. AND ITS CONFIGURATION DATA SET COMPLETED

 INPUT OF A T.F. AND ITS CONFIGURATION DATA SET COMPLETED

 DATA TRANSMISSION SUCCESSFUL

 ENTER 1 ** IF PLOT IS DESIRED
       0 ** OTHERWISE

1 **
 ENTER 1 ** FOR PLOTTING ALL YS IN THE STANDARD FORM
       0 ** OTHERWISE

0 **
 ENTER ID NO. FOR DESIRED VARIABLE Y

           ID         Y

           1          1
           2          2
           3          3
3 **
 ENTER 1 ** FOR STANDARD GRAPH
       0 ** OTHERWISE

0 **
 ENTER ID NO. AND NEW VALUES **

  1  YMIN,YMAX...................................**
  2  PLOT STATUS..1=NEW,2=OVERTOP,3=OVERBOTTOM..**
  3  PLOT TYPE..-VE=LINE,0=+,1=X ETC............**
  4  DEVICE..1=SCOPE,2=PLOTTER..................**
  5  PLOT POSITION..1=TOP,2=BOTTOM..............**
  6 **..LABEL ON X AXIS...............*..(20A1)..*
  7 **..LABEL ON Y AXIS...............*..(20A1)..*
  8 **..TITLE.........................*..(40A1)..*

 LAST ENTRY IS 999 **

1 0.0 2.5 **
5 2 **
6 **
*    TIME ANY UNIT    *
8 **
*   COMPARISON OF RESPONSES                      *
999 **
```

```
ENTER 1 ** IF PLOT IS DESIRED
      0 ** OTHERWISE

0 **
ENTER NOP(J) VALUE FOR FURTHER CHANGES
      0 ** OTHERWISE

3 **
ENTER T.F.TYPE OR PARAMETER **
ENTER NOP(J) VALUE TO STOP OR MAKE A DIFFERENT CHANGE

502 2 1 **
502 2 1 **

999 0 **
999 0 **

END OF INPUT DATA FOR JOB NUMBER     1

INPUT OF A T.F. AND ITS CONFIGURATION DATA SET COMPLETED

INPUT OF A T.F. AND ITS CONFIGURATION DATA SET COMPLETED

DATA TRANSMISSION SUCCESSFUL
```

```
                                             * JULY 69 *

                  * IN THIS EXAMPLE PI CONTROLLER CONSTANTS *
                  * FROM LIBRARY ARE COMPARED FOR A UNIT    *
                  * STEP CHANGE IN LOAD                     *
                  * RATIO OF TIME DELAY TO TIME LAG IS 0.8  *
                  * PROCESS GAIN IS 10                      *

        * INPUT DATA CARDS *

* THIS IS THE MATRIX INPUT FLAG *
0                            **
* THIS IS TO OBTAIN A PLOT OF TIME DOMAIN DATA *
0 0 1 8 3                    **
* THIS IS THE NUMBER OF TIME INTERVALS =50 *
0 0 3 50                     **
* THIS IS TO PROCEED FOR POSITIVE EIGENVALUES*
0 0 1 6 1                    **
1 0 1.0 0.0                  **
101 101 3 1.0 -1.0           **
203 10.0 0.4 0.5             **
103 2 102 1.0 1.0            **
502 7 2                      **
102 1 1.0                    **
* THIS IS THE FORCING FUNCTION *
999 101 1 0.0 0.0            **
999 102 1 1.0 0.0            **
* THIS IS TO CHANGE THE LOGICAL UNIT NUMBERS *
0 3 4 **
```

CONTROLLER CONSTANTS FROM LIBRARY

```
PROCESS GAIN...............................    10.0000
TIME DELAY.................................     0.4000
TIME LAG...................................     0.5000
CONTROLLER CODE............................          5
CODED CRITERION TO SELECT CONSTANTS........          7
CODED DISTURBANCE TO SELECT CONSTANTS......          2

PROPORTIONAL CONSTANT......................     0.0875
INTEGRAL TIME..............................     0.9333
DERIVATIVE TIME............................     0.0000
```

BLOCK DIAGRAM INPUT

```
INPUT MATRIX FLAG..........................          0
JOB NUMBER.................................          1
PRIORITY NUMBER FOR PRINTING OUTPUT........          3
```

```
LOGICAL UNIT NUMBER FOR INPUT DEVICE....            3
LOGICAL UNIT NUMBER FOR OUTPUT DEVICE...            4

OPTION.. 1..............................            0
OPTION.. 2..............................            0
OPTION.. 3..............................            0
OPTION.. 4..............................            2
OPTION.. 5..............................            0
OPTION.. 6..............................            1
OPTION.. 7..............................            2
OPTION.. 8..............................            3
OPTION.. 9..............................            0
OPTION..10..............................            0

ABSOLUTE CRITERION VALUE (PER CENT).....       0.1000
INITIAL TIME VALUE......................       0.0000
TIME INTERVAL...........................       0.1000
TIME SUBINTERVAL........................       0.0100
NUMBER OF TIME INTERVALS................           50

INITIAL STATE VARIABLE.. 1..............       0.0000
INITIAL STATE VARIABLE.. 2..............       0.0000

DESIRED OUTPUT/STATE VARIABLE.. 1.......            1
DESIRED OUTPUT/STATE VARIABLE.. 2.......            2
DESIRED OUTPUT/STATE VARIABLE.. 3.......            3

UTILITY PROGRAM NUMBER..................            0
NUMBER OF JOBS..........................            1

DIST.. 1  CODE.. 1  1ST PAR..    0.0000  2ND PAR..     0.0000
DIST.. 2  CODE.. 1  1ST PAR..    1.0000  2ND PAR..     0.0000
```

COEFFICIENT MATRICES OF THE STATE EQUATION
AND RELATED DATA

```
NUMBER OF STATE VARIABLES...............            2
NUMBER OF OUTPUT VARIABLES..............            3
NUMBER OF EXTERNAL FORCING FUNCTIONS....            2
NUMBER OF TIME DELAYS...................            1

NUMBER OF EXTERNAL FORCING FUNCTIONS IN U(T)        1
DELAYED EXTERNAL FORCING FUNCTIONS IN...U(T)        2
DELAYED STATE VARIABLES IN..............U(T)        2

VARIABLE IN U(T).... 1...DELAY VALUE....       0.0000
VARIABLE IN U(T).... 1...DELAY VALUE....       0.4000
VARIABLE IN U(T).... 2...DELAY VALUE....       0.4000
VARIABLE IN U(T).... 1...DELAY VALUE....       0.4000
VARIABLE IN U(T).... 2...DELAY VALUE....       0.4000
```

NUMBER OF EXTERNAL FORCING FUNCTIONS IN V(T) 1

VARIABLE IN V(T).... 1...DELAY VALUE.... 0.0000

MATRIX A

-2.0000	0.0000
-0.0937	0.0000

MATRIX B

0.0000	1.7499	20.0000	-1.7499	20.0000
0.0937	0.0000	0.0000	0.0000	0.0000

MATRIX C

-1.0000	0.0000
-0.0874	1.0000
1.0000	0.0000

MATRIX D

1.0000
0.0874
0.0000

COEFFICIENT MATRICES OF THE STATE
DIFFERENCE EQUATION

MATRIX FM(T)

0.8187E 00	0.0000E 00
-0.8497E-02	0.1000E 01

MATRIX H(T)

0.0000E 00	0.1586E 00	0.1812E 01	-0.1586E 00	0.1812E 01
0.9375E-02	-0.7680E-03	-0.8777E-02	0.7680E-03	-0.8777E-02

TIME DOMAIN RESPONSE

TIME	Y(1)	Y(2)	Y(3)
0.0000	0.0000	0.0000	0.0000
0.1000	0.0000	0.0000	0.0000
0.2000	0.0000	0.0000	0.0000
0.3000	0.0000	0.0000	0.0000
0.4000	0.0000	0.0000	0.0000
0.5000	-1.8127	-0.1673	1.8127
0.6000	-3.2969	-0.3214	3.2969
0.7000	-4.5120	-0.4645	4.5120
0.8000	-5.5068	-0.5987	5.5068
0.9000	-6.3214	-0.7255	6.3214
1.0000	-6.6848	-0.8183	6.6848
1.1000	-6.7031	-0.8827	6.7031
1.2000	-6.4587	-0.9230	6.4587
1.3000	-6.0153	-0.9426	6.0153
1.4000	-5.4224	-0.9442	5.4224
1.5000	-4.7687	-0.9347	4.7687
1.6000	-4.1169	-0.9192	4.1169
1.7000	-3.5102	-0.9018	3.5102
1.8000	-2.9779	-0.8855	2.9779
1.9000	-2.5391	-0.8729	2.5391
2.0000	-2.1972	-0.8651	2.1972
2.1000	-1.9453	-0.8625	1.9453
2.2000	-1.7707	-0.8646	1.7707
2.3000	-1.6571	-0.8707	1.6571
2.4000	-1.5870	-0.8798	1.5870
2.5000	-1.5437	-0.8906	1.5437
2.6000	-1.5131	-0.9023	1.5131
2.7000	-1.4841	-0.9138	1.4841
2.8000	-1.4494	-0.9245	1.4494
2.9000	-1.4045	-0.9340	1.4045
3.0000	-1.3480	-0.9419	1.3480
3.1000	-1.2807	-0.9483	1.2807
3.2000	-1.2047	-0.9533	1.2047
3.3000	-1.1231	-0.9571	1.1231
3.4000	-1.0391	-0.9598	1.0391
3.5000	-0.9560	-0.9619	0.9560
3.6000	-0.8763	-0.9635	0.8763
3.7000	-0.8019	-0.9649	0.8019
3.8000	-0.7343	-0.9661	0.7343
3.9000	-0.6739	-0.9674	0.6739
4.0000	-0.6207	-0.9688	0.6207
4.1000	-0.5742	-0.9704	0.5742
4.2000	-0.5337	-0.9720	0.5337
4.3000	-0.4983	-0.9737	0.4983
4.4000	-0.4669	-0.9755	0.4669
4.5000	-0.4386	-0.9773	0.4386
4.6000	-0.4127	-0.9790	0.4127
4.7000	-0.3885	-0.9806	0.3885

4.8000	-0.3656	-0.9822	0.3656
4.9000	-0.3436	-0.9836	0.3436
5.0000	-0.3224	-0.9848	0.3224

```
                                                  * JULY 69 *

            * THIS EXAMPLE DEALS WITH ZIEGLER-NICHOLS     *
            * CLOSED-LOOP TUNING.THE SYSTEM IS SUBJECTED *
            * TO A STEP CHANGE IN SET POINT.THE GAIN IS   *
            * INCREASED TILL SUSTAINED OSCILLATIONS       *
            * EXAMPLE 19.1,PAGE 237 OF COUGHANOWR-KOPPEL *

       * INPUT DATA CARDS *

* THIS IS THE MATRIX INPUT FLAG *
0                           **
* THIS DEFINES 50 TIME INTERVALS *
0 0 3 50                    **
401 3.0                     **
2 0 1.0 0.0 -1 1. 1. -1 1. 2.0  **
* THIS IS AN ISOLATED TIME DELAY *
3 0 1.0 0.5                 **
101 101 3 1.0 -1.0          **
102 1 1.0                   **
103 2 1.0                   **
999 101 1 1.0 0.0           **
* THIS IS TO OBTAIN A PLOT OF TIME DOMAIN RESPONSE *
0 0 1 8 3                   **
* THIS CARD DEFINES NEW LOGICAL UNIT NUMBERS *
0 22 3 3 4                  **
```

BLOCK DIAGRAM INPUT

```
INPUT MATRIX FLAG.......................              0
JOB NUMBER..............................             22
PRIORITY NUMBER FOR PRINTING OUTPUT.....              3
LOGICAL UNIT NUMBER FOR INPUT DEVICE....              3
LOGICAL UNIT NUMBER FOR OUTPUT DEVICE...              4

OPTION.. 1..............................              0
OPTION.. 2..............................              0
OPTION.. 3..............................              0
OPTION.. 4..............................              2
OPTION.. 5..............................              0
OPTION.. 6..............................              0
OPTION.. 7..............................              2
OPTION.. 8..............................              3
OPTION.. 9..............................              0
OPTION..10..............................              0

ABSOLUTE CRITERION VALUE (PER CENT).....         0.1000
INITIAL TIME VALUE......................         0.0000
TIME INTERVAL...........................         0.1000
```

```
TIME SUBINTERVAL.........................    0.0100
NUMBER OF TIME INTERVALS.................        50

INITIAL STATE VARIABLE.. 1...............    0.0000
INITIAL STATE VARIABLE.. 2...............    0.0000

DESIRED OUTPUT/STATE VARIABLE.. 1.......          1
DESIRED OUTPUT/STATE VARIABLE.. 2.......          2
DESIRED OUTPUT/STATE VARIABLE.. 3.......          3

UTILITY PROGRAM NUMBER..................          0
NUMBER OF JOBS..........................          1

DIST.. 1  CODE.. 1  1ST PAR..    1.0000  2ND PAR..    0.0000
```

COEFFICIENT MATRICES OF THE STATE EQUATION
AND RELATED DATA

```
NUMBER OF STATE VARIABLES...............          2
NUMBER OF OUTPUT VARIABLES..............          3
NUMBER OF EXTERNAL FORCING FUNCTIONS....          1
NUMBER OF TIME DELAYS...................          1

NUMBER OF EXTERNAL FORCING FUNCTIONS IN U(T)      1
DELAYED EXTERNAL FORCING FUNCTIONS IN...U(T)      0
DELAYED STATE VARIABLES IN..............U(T)      1

VARIABLE IN U(T).... 1...DELAY VALUE....    0.0000
VARIABLE IN U(T).... 1...DELAY VALUE....    0.5000

NUMBER OF EXTERNAL FORCING FUNCTIONS IN V(T)      1
DELAYED EXTERNAL FORCING FUNCTIONS IN...V(T)      0
DELAYED STATE VARIABLES IN..............V(T)      1

VARIABLE IN V(T).... 1...DELAY VALUE....    0.0000
VARIABLE IN V(T).... 1...DELAY VALUE....    0.5000
```

MATRIX A

```
 0.0000    1.0000

-0.5000   -1.5000
```

MATRIX B

```
 0.0000    0.0000

 1.5000   -1.5000
```

MATRIX C

0.0000	0.0000
1.0000	0.0000
0.0000	0.0000

MATRIX D

3.0000	-3.0000
0.0000	0.0000
0.0000	1.0000

COEFFICIENT MATRICES OF THE STATE
DIFFERENCE EQUATION

MATRIX FM(T)

0.9976E 00	0.9278E-01
-0.4639E-01	0.8584E 00

MATRIX H(T)

0.7133E-02	-0.7133E-02
0.1391E 00	-0.1391E 00

TIME DOMAIN RESPONSE

TIME	Y(1)	Y(2)	Y(3)
0.0000	3.0000	0.0000	0.0000
0.1000	3.0000	0.0071	0.0000
0.2000	3.0000	0.0271	0.0000
0.3000	3.0000	0.0582	0.0000
0.4000	3.0000	0.0985	0.0000
0.5000	3.0000	0.1467	0.0000
0.6000	2.9785	0.2015	0.0071
0.7000	2.9185	0.2615	0.0271
0.8000	2.8253	0.3257	0.0582
0.9000	2.7042	0.3927	0.0985
1.0000	2.5596	0.4614	0.1467
1.1000	2.3954	0.5306	0.2015

1.2000	2.2152	0.5992	0.2615
1.3000	2.0228	0.6661	0.3257
1.4000	1.8217	0.7304	0.3927
1.5000	1.6156	0.7913	0.4614
1.6000	1.4080	0.8481	0.5306
1.7000	1.2023	0.8999	0.5992
1.8000	1.0015	0.9465	0.6661
1.9000	0.8085	0.9872	0.7304
2.0000	0.6258	1.0219	0.7913
2.1000	0.4556	1.0503	0.8481
2.2000	0.3000	1.0724	0.8999
2.3000	0.1604	1.0882	0.9465
2.4000	0.0382	1.0978	0.9872
2.5000	-0.0658	1.1014	1.0219
2.6000	-0.1510	1.0993	1.0503
2.7000	-0.2173	1.0919	1.0724
2.8000	-0.2647	1.0796	1.0882
2.9000	-0.2934	1.0629	1.0978
3.0000	-0.3043	1.0423	1.1014
3.1000	-0.2981	1.0184	1.0993
3.2000	-0.2759	0.9916	1.0919
3.3000	-0.2390	0.9626	1.0796
3.4000	-0.1889	0.9321	1.0629
3.5000	-0.1271	0.9004	1.0423
3.6000	-0.0552	0.8683	1.0184
3.7000	0.0250	0.8362	0.9916
3.8000	0.1119	0.8046	0.9626
3.9000	0.2036	0.7740	0.9321
4.0000	0.2986	0.7448	0.9004
4.1000	0.3950	0.7174	0.8683
4.2000	0.4913	0.6920	0.8362
4.3000	0.5861	0.6691	0.8046
4.4000	0.6779	0.6488	0.7740
4.5000	0.7655	0.6312	0.7448
4.6000	0.8477	0.6165	0.7174
4.7000	0.9237	0.6047	0.6920
4.8000	0.9924	0.5959	0.6691
4.9000	1.0535	0.5900	0.6488
5.0000	1.1062	0.5869	0.6312

```
                                                     * JULY 69 *

          * THIS IS A THIRD ORDER EXAMPLE TO TEST THE   *
          * CAYLEY-HAMILTON TECHNIQUE AGAINST THE POWER*
          * SERIES METHOD.THE EXAMPLE HAS BEEN CHOSEN   *
          * BECAUSE OF COMPLEX EIGENVALUES              *

     * INPUT DATA CARDS *

* THIS IS THE MATRIX INPUT FLAG *
0                          **
* THIS GIVES THE NUMBER OF TIME INTERVALS *
0 0 3 50                   **
* THIS IS FOR PLOTTING THE TIME DOMAIN RESPONSE *
0 0 1 8 3                  **
401 1.0                    **
402 2.0                    **
3 0 1.0 0.2 -2 8.0 0.2 1.0   **
204 1.0 0.0 1.0            **
101 101 4 1.0 -1.0         **
102 1 1.0                  **
103 2 1.0                  **
104 3 1.0                  **
999 101 1 1.0 0.0          **
0 3 4                      **
```

BLOCK DIAGRAM INPUT

```
INPUT MATRIX FLAG........................          0
JOB NUMBER...............................          1
PRIORITY NUMBER FOR PRINTING OUTPUT......          3
LOGICAL UNIT NUMBER FOR INPUT DEVICE.....          3
LOGICAL UNIT NUMBER FOR OUTPUT DEVICE....          4

OPTION.. 1...............................          0
OPTION.. 2...............................          0
OPTION.. 3...............................          0
OPTION.. 4...............................          2
OPTION.. 5...............................          0
OPTION.. 6...............................          0
OPTION.. 7...............................          2
OPTION.. 8...............................          3
OPTION.. 9...............................          0
OPTION..10...............................          0

ABSOLUTE CRITERION VALUE (PER CENT)......     0.1000
INITIAL TIME VALUE.......................     0.0000
TIME INTERVAL............................     0.1000
TIME SUBINTERVAL.........................     0.0100
```

```
NUMBER OF TIME INTERVALS................          50

INITIAL STATE VARIABLE.. 1..............      0.0000
INITIAL STATE VARIABLE.. 2..............      0.0000
INITIAL STATE VARIABLE.. 3..............      0.0000

DESIRED OUTPUT/STATE VARIABLE.. 1.......           1
DESIRED OUTPUT/STATE VARIABLE.. 2.......           2
DESIRED OUTPUT/STATE VARIABLE.. 3.......           3
DESIRED OUTPUT/STATE VARIABLE.. 4.......           4

UTILITY PROGRAM NUMBER..................           0
NUMBER OF JOBS..........................           1

DIST.. 1  CODE.. 1  1ST PAR..    1.0000  2ND PAR..    0.0000
```

COEFFICIENT MATRICES OF THE STATE EQUATION
AND RELATED DATA

```
NUMBER OF STATE VARIABLES...............           3
NUMBER OF OUTPUT VARIABLES..............           4
NUMBER OF EXTERNAL FORCING FUNCTIONS....           1
NUMBER OF TIME DELAYS...................           1

NUMBER OF EXTERNAL FORCING FUNCTIONS IN U(T)       0
DELAYED EXTERNAL FORCING FUNCTIONS IN...U(T)       1
DELAYED STATE VARIABLES IN..............U(T)       1

VARIABLE IN U(T).... 1...DELAY VALUE....      0.2000
VARIABLE IN U(T).... 3...DELAY VALUE....      0.2000

NUMBER OF EXTERNAL FORCING FUNCTIONS IN V(T)       1

VARIABLE IN V(T).... 1...DELAY VALUE....      0.0000
```

MATRIX A

```
   0.0000    1.0000    0.0000

  -8.0000   -0.2000    0.0000

   1.0000    0.0000   -1.0000
```

MATRIX B

```
   0.0000    0.0000

   2.0000   -2.0000
```

0.0000	0.0000

MATRIX C

0.0000	0.0000	-1.0000
0.0000	0.0000	-2.0000
1.0000	0.0000	0.0000
0.0000	0.0000	1.0000

MATRIX D

1.0000
2.0000
0.0000
0.0000

COEFFICIENT MATRICES OF THE STATE
DIFFERENCE EQUATION

MATRIX FM(T)

0.9605E 00	0.9769E-01	0.0000E 00
-0.7815E 00	0.9409E 00	0.0000E 00
0.9387E-01	0.4772E-02	0.9048E 00

MATRIX H(T)

0.9866E-02	-0.9866E-02
0.1953E 00	-0.1953E 00
0.3237E-03	-0.3237E-03

TIME DOMAIN RESPONSE

TIME	Y(1)	Y(2)	Y(3)	Y(4)
0.0000	1.0000	2.0000	0.0000	0.0000
0.1000	1.0000	2.0000	0.0000	0.0000

0.2000	1.0000	2.0000	0.0000	0.0000
0.3000	0.9996	1.9993	0.0098	0.0003
0.4000	0.9975	1.9950	0.0384	0.0024
0.5000	0.9920	1.9841	0.0830	0.0079
0.6000	0.9822	1.9644	0.1399	0.0177
0.7000	0.9675	1.9350	0.2043	0.0324
0.8000	0.9479	1.8958	0.2709	0.0520
0.9000	0.9239	1.8479	0.3343	0.0760
1.0000	0.8966	1.7933	0.3893	0.1033
1.1000	0.8672	1.7345	0.4314	0.1327
1.2000	0.8374	1.6749	0.4571	0.1625
1.3000	0.8089	1.6178	0.4643	0.1910
1.4000	0.7833	1.5667	0.4521	0.2166
1.5000	0.7622	1.5245	0.4214	0.2377
1.6000	0.7469	1.4939	0.3743	0.2530
1.7000	0.7382	1.4764	0.3143	0.2617
1.8000	0.7364	1.4729	0.2460	0.2635
1.9000	0.7416	1.4832	0.1746	0.2583
2.0000	0.7529	1.5059	0.1054	0.2470
2.1000	0.7694	1.5389	0.0438	0.2305
2.2000	0.7897	1.5795	-0.0054	0.2102
2.3000	0.8120	1.6240	-0.0386	0.1879
2.4000	0.8344	1.6689	-0.0532	0.1655
2.5000	0.8551	1.7103	-0.0479	0.1448
2.6000	0.8724	1.7449	-0.0232	0.1275
2.7000	0.8849	1.7698	0.0190	0.1150
2.8000	0.8914	1.7828	0.0755	0.1085
2.9000	0.8913	1.7827	0.1419	0.1086
3.0000	0.8847	1.7695	0.2132	0.1152
3.1000	0.8720	1.7440	0.2837	0.1279
3.2000	0.8540	1.7080	0.3480	0.1459
3.3000	0.8320	1.6641	0.4012	0.1679
3.4000	0.8079	1.6158	0.4390	0.1920
3.5000	0.7833	1.5666	0.4585	0.2166
3.6000	0.7601	1.5203	0.4581	0.2398
3.7000	0.7402	1.4804	0.4377	0.2597
3.8000	0.7250	1.4500	0.3990	0.2749
3.9000	0.7157	1.4314	0.3449	0.2842
4.0000	0.7130	1.4260	0.2794	0.2869
4.1000	0.7172	1.4344	0.2077	0.2827
4.2000	0.7278	1.4557	0.1351	0.2721
4.3000	0.7442	1.4884	0.0675	0.2557
4.4000	0.7650	1.5300	0.0099	0.2349
4.5000	0.7886	1.5772	-0.0330	0.2113
4.6000	0.8132	1.6265	-0.0580	0.1867
4.7000	0.8369	1.6739	-0.0632	0.1630
4.8000	0.8579	1.7158	-0.0480	0.1420
4.9000	0.8745	1.7490	-0.0136	0.1254
5.0000	0.8853	1.7707	0.0374	0.1146

* JULY 69 *

```
* THIS EXAMPLE CONSISTS OF A MULTILOOP    *
* FEEDBACK BLOCK DIAGRAM.IT IS GIVEN AS   *
* PROBLEM A-4-1 PAGE 223 IN OGATA.THE OUT-*
* PUT IS AN EXAMPLE OF LEVEL 5 OF DOCUMEN-*
* TATION(SEE USER' MANUAL)                *
```

* INPUT DATA CARDS *

```
0                            **
0 0 3 50                     **
0 0 2 10. 0.0 0.1 0.01       **
0 0 1 6 1                    **

3 0 1.0 0.0                  **
5 1 0.0 1.0                  **
2 0 10.0 0.0 -1 3.0 1.0       **
4 0 2.0 0.0 -1 0.0 1.0 -1 1.0 1.0     **
1 0 1.0 0.0                  **
101 101 4 1.0 -1.0           **
102 1 1.0                    **
103 2 5 1.0 -1.0             **
104 3 1.0                    **
105 4 1.0                    **
999 101 1 1.0 0.0 **
0 0 1 8 3                    **
0 5 5 3 4                    **
```

BLOCK DIAGRAM INPUT

```
INPUT MATRIX FLAG.......................         0
JOB NUMBER..............................         5
PRIORITY NUMBER FOR PRINTING OUTPUT.....         5
LOGICAL UNIT NUMBER FOR INPUT DEVICE....         3
LOGICAL UNIT NUMBER FOR OUTPUT DEVICE...         4

OPTION.. 1..............................         0
OPTION.. 2..............................         0
OPTION.. 3..............................         0
OPTION.. 4..............................         2
OPTION.. 5..............................         0
OPTION.. 6..............................         1
OPTION.. 7..............................         2
OPTION.. 8..............................         3
OPTION.. 9..............................         0
OPTION..10..............................         0

ABSOLUTE CRITERION VALUE (PER CENT).....    0.0000
```

```
INITIAL TIME VALUE.......................     0.0000
TIME INTERVAL............................     0.1000
TIME SUBINTERVAL.........................     0.0100
NUMBER OF TIME INTERVALS.................         50

INITIAL STATE VARIABLE.. 1...............     0.0000
INITIAL STATE VARIABLE.. 2...............     0.0000
INITIAL STATE VARIABLE.. 3...............     0.0000

DESIRED OUTPUT/STATE VARIABLE.. 1........          1
DESIRED OUTPUT/STATE VARIABLE.. 2........          2
DESIRED OUTPUT/STATE VARIABLE.. 3........          3
DESIRED OUTPUT/STATE VARIABLE.. 4........          4
DESIRED OUTPUT/STATE VARIABLE.. 5........          5

UTILITY PROGRAM NUMBER...................          0
NUMBER OF JOBS...........................          1

DIST.. 1  CODE.. 1  1ST PAR..    1.0000  2ND PAR..    0.0000
```

COEFFICIENT MATRICES OF THE STATE EQUATION
AND RELATED DATA

```
NUMBER OF STATE VARIABLES................          3
NUMBER OF OUTPUT VARIABLES...............          5
NUMBER OF EXTERNAL FORCING FUNCTIONS.....          1
NUMBER OF TIME DELAYS....................          0

NUMBER OF EXTERNAL FORCING FUNCTIONS IN U(T)       1

VARIABLE IN U(T).... 1...DELAY VALUE....     0.0000

NUMBER OF EXTERNAL FORCING FUNCTIONS IN V(T)       1

VARIABLE IN V(T).... 1...DELAY VALUE....     0.0000
```

MATRIX A

```
-3.0000  -10.0000   0.0000

 0.0000    0.0000   1.0000

 2.0000    0.0000  -3.0000
```

MATRIX B

```
10.0000

 0.0000
```

```
0.0000

                         MATRIX C

0.0000    -1.0000     0.0000

1.0000     0.0000     0.0000

1.0000     0.0000    -1.0000

0.0000     1.0000     0.0000

0.0000     0.0000     1.0000

                         MATRIX D

1.0000

0.0000

0.0000

0.0000

0.0000

COEFFICIENT OF POWER... 0  OF CHAR. EQUATION......  0.1999E 02
COEFFICIENT OF POWER... 1  OF CHAR. EQUATION......  0.8999E 01
COEFFICIENT OF POWER... 2  OF CHAR. EQUATION......  0.6000E 01
COEFFICIENT OF POWER... 3  OF CHAR. EQUATION......  0.1000E 01

REAL PART OF EI.V. -0.4999E 00   IMAG. PART OF EI.V.  0.1936E 01
REAL PART OF EI.V. -0.4999E 00   IMAG. PART OF EI.V. -0.1936E 01
REAL PART OF EI.V. -0.5000E 01   IMAG. PART OF EI.V.  0.0000E 00

REAL PART OF SHIFTED EI.V. -0.4500E 01   IMAG. PART.  0.0000E 00
REAL PART OF SHIFTED EI.V.  0.0000E 00   IMAG. PART. -0.1936E 01
REAL PART OF SHIFTED EI.V.  0.0000E 00   IMAG. PART.  0.1936E 01

REAL PART OF SMALLEST EI.V........................ -0.4999E 00

REAL PART OF ORDERED EI.V. -0.4500E 01   IMAG. PART.  0.0000E 00
REAL PART OF ORDERED EI.V.  0.0000E 00   IMAG. PART.  0.1936E 01
REAL PART OF ORDERED EI.V.  0.0000E 00   IMAG. PART. -0.1936E 01

NUMBER OF DISTINCT EI.VS..........................        1
NUMBER OF REPEATED EI.VS..........................        0
NUMBER OF COMPLEX EI.VS...........................        2

                         MATRIX CC
```

```
  0.1000E 01 -0.4999E 01  0.2499E 02

  0.1000E 01 -0.4999E 00 -0.3499E 01

  0.0000E 00  0.1936E 01 -0.1936E 01


TIME.....    0.0000  EXPONENTIAL OF EI.V( 1).......  0.1000E 01
TIME.....    0.0000  EXPONENTIAL OF EI.V( 2).......  0.1000E 01
TIME.....    0.0000  EXPONENTIAL OF EI.V( 3).......  0.0000E 00

TIME.....    0.0000  COEFFICIENT ALPHA( 0).........  0.9999E 00
TIME.....    0.0000  COEFFICIENT ALPHA( 1).........  0.4967E-08
TIME.....    0.0000  COEFFICIENT ALPHA( 2).........  0.4967E-08

             FUNDAMENTAL MATRIX AT T =     0.0000

  0.9999E 00  0.9934E-07 -0.4967E-07

  0.9934E-08  0.9999E 00 -0.9934E-08

 -0.4967E-07 -0.9934E-07  0.9999E 00


TIME.....    0.0100  EXPONENTIAL OF EI.V( 1).......  0.9559E 00
TIME.....    0.0100  EXPONENTIAL OF EI.V( 2).......  0.9998E 00
TIME.....    0.0100  EXPONENTIAL OF EI.V( 3).......  0.1936E-01

TIME.....    0.0100  COEFFICIENT ALPHA( 0).........  0.1005E 01
TIME.....    0.0100  COEFFICIENT ALPHA( 1).........  0.1004E-01
TIME.....    0.0100  COEFFICIENT ALPHA( 2).........  0.4925E-04

             FUNDAMENTAL MATRIX AT T =     0.0100

  0.9704E 00 -0.9851E-01 -0.4901E-03

  0.9802E-04  0.9999E 00  0.9851E-02

  0.1940E-01 -0.9802E-03  0.9704E 00


TIME.....    0.0200  EXPONENTIAL OF EI.V( 1).......  0.9139E 00
TIME.....    0.0200  EXPONENTIAL OF EI.V( 2).......  0.9992E 00
TIME.....    0.0200  EXPONENTIAL OF EI.V( 3).......  0.3872E-01

TIME.....    0.0200  COEFFICIENT ALPHA( 0).........  0.1010E 01
TIME.....    0.0200  COEFFICIENT ALPHA( 1).........  0.2018E-01
TIME.....    0.0200  COEFFICIENT ALPHA( 2).........  0.1941E-03
```

FUNDAMENTAL MATRIX AT T = 0.0200

```
0.9417E 00 -0.1941E 00 -0.1921E-02

0.3843E-03  0.9999E 00  0.1941E-01

0.3767E-01 -0.3843E-02  0.9417E 00
```

```
TIME.....    0.0300  EXPONENTIAL OF EI.V( 1).......  0.8737E 00
TIME.....    0.0300  EXPONENTIAL OF EI.V( 2).......  0.9983E 00
TIME.....    0.0300  EXPONENTIAL OF EI.V( 3).......  0.5806E-01

TIME.....    0.0300  COEFFICIENT ALPHA( 0).........  0.1015E 01
TIME.....    0.0300  COEFFICIENT ALPHA( 1).........  0.3041E-01
TIME.....    0.0300  COEFFICIENT ALPHA( 2).........  0.4302E-03
```

FUNDAMENTAL MATRIX AT T = 0.0300

```
0.9138E 00 -0.2868E 00 -0.4238E-02

0.8477E-03  0.9999E 00  0.2868E-01

0.5483E-01 -0.8477E-02  0.9138E 00
```

```
TIME.....    0.0400  EXPONENTIAL OF EI.V( 1).......  0.8352E 00
TIME.....    0.0400  EXPONENTIAL OF EI.V( 2).......  0.9970E 00
TIME.....    0.0400  EXPONENTIAL OF EI.V( 3).......  0.7738E-01

TIME.....    0.0400  COEFFICIENT ALPHA( 0).........  0.1019E 01
TIME.....    0.0400  COEFFICIENT ALPHA( 1).........  0.4071E-01
TIME.....    0.0400  COEFFICIENT ALPHA( 2).........  0.7537E-03
```

FUNDAMENTAL MATRIX AT T = 0.0400

```
0.8867E 00 -0.3769E 00 -0.7387E-02

0.1477E-02  0.9997E 00  0.3769E-01

0.7094E-01 -0.1477E-01  0.8867E 00
```

```
TIME.....    0.0500  EXPONENTIAL OF EI.V( 1).......  0.7985E 00
TIME.....    0.0500  EXPONENTIAL OF EI.V( 2).......  0.9953E 00
TIME.....    0.0500  EXPONENTIAL OF EI.V( 3).......  0.9667E-01
```

```
TIME.....    0.0500  COEFFICIENT ALPHA( 0).........  0.1024E 01
TIME.....    0.0500  COEFFICIENT ALPHA( 1).........  0.5108E-01
TIME.....    0.0500  COEFFICIENT ALPHA( 2).........  0.1160E-02

           FUNDAMENTAL MATRIX AT T =    0.0500

  0.8603E 00 -0.4642E 00 -0.1131E-01

  0.2263E-02  0.9996E 00  0.4642E-01

  0.8606E-01 -0.2263E-01  0.8603E 00

TIME.....    0.0600  EXPONENTIAL OF EI.V( 1).......  0.7633E 00
TIME.....    0.0600  EXPONENTIAL OF EI.V( 2).......  0.9932E 00
TIME.....    0.0600  EXPONENTIAL OF EI.V( 3).......  0.1159E 00

TIME.....    0.0600  COEFFICIENT ALPHA( 0).........  0.1029E 01
TIME.....    0.0600  COEFFICIENT ALPHA( 1).........  0.6151E-01
TIME.....    0.0600  COEFFICIENT ALPHA( 2).........  0.1646E-02

           FUNDAMENTAL MATRIX AT T =    0.0600

  0.8346E 00 -0.5490E 00 -0.1597E-01

  0.3195E-02  0.9993E 00  0.5490E-01

  0.1002E 00 -0.3195E-01  0.8346E 00

TIME.....    0.0700  EXPONENTIAL OF EI.V( 1).......  0.7297E 00
TIME.....    0.0700  EXPONENTIAL OF EI.V( 2).......  0.9908E 00
TIME.....    0.0700  EXPONENTIAL OF EI.V( 3).......  0.1351E 00

TIME.....    0.0700  COEFFICIENT ALPHA( 0).........  0.1034E 01
TIME.....    0.0700  COEFFICIENT ALPHA( 1).........  0.7199E-01
TIME.....    0.0700  COEFFICIENT ALPHA( 2).........  0.2208E-02

           FUNDAMENTAL MATRIX AT T =    0.0700

  0.8096E 00 -0.6312E 00 -0.2132E-01

  0.4264E-02  0.9989E 00  0.6312E-01

  0.1134E 00 -0.4264E-01  0.8096E 00

TIME.....    0.0800  EXPONENTIAL OF EI.V( 1).......  0.6976E 00
```

```
TIME.....    0.0800  EXPONENTIAL OF EI.V( 2).......  0.9880E 00
TIME.....    0.0800  EXPONENTIAL OF EI.V( 3).......  0.1543E 00

TIME.....    0.0800  COEFFICIENT ALPHA( 0).........  0.1039E 01
TIME.....    0.0800  COEFFICIENT ALPHA( 1).........  0.8252E-01
TIME.....    0.0800  COEFFICIENT ALPHA( 2).........  0.2842E-02

              FUNDAMENTAL MATRIX AT T =    0.0800

  0.7852E 00 -0.7109E 00 -0.2730E-01

  0.5461E-02  0.9984E 00  0.7109E-01

  0.1258E 00 -0.5461E-01  0.7852E 00

TIME.....    0.0900  EXPONENTIAL OF EI.V( 1).......  0.6669E 00
TIME.....    0.0900  EXPONENTIAL OF EI.V( 2).......  0.9848E 00
TIME.....    0.0900  EXPONENTIAL OF EI.V( 3).......  0.1734E 00

TIME.....    0.0900  COEFFICIENT ALPHA( 0).........  0.1043E 01
TIME.....    0.0900  COEFFICIENT ALPHA( 1).........  0.9308E-01
TIME.....    0.0900  COEFFICIENT ALPHA( 2).........  0.3544E-02

              FUNDAMENTAL MATRIX AT T =    0.0900

  0.7613E 00 -0.7882E 00 -0.3388E-01

  0.6777E-02  0.9978E 00  0.7882E-01

  0.1373E 00 -0.6777E-01  0.7613E 00

TIME.....    0.1000  EXPONENTIAL OF EI.V( 1).......  0.6376E 00
TIME.....    0.1000  EXPONENTIAL OF EI.V( 2).......  0.9813E 00
TIME.....    0.1000  EXPONENTIAL OF EI.V( 3).......  0.1924E 00

TIME.....    0.1000  COEFFICIENT ALPHA( 0).........  0.1048E 01
TIME.....    0.1000  COEFFICIENT ALPHA( 1).........  0.1036E 00
TIME.....    0.1000  COEFFICIENT ALPHA( 2).........  0.4313E-02

              FUNDAMENTAL MATRIX AT T =    0.1000

  0.7381E 00 -0.8632E 00 -0.4102E-01

  0.8205E-02  0.9971E 00  0.8632E-01

  0.1480E 00 -0.8205E-01  0.7381E 00
```

COEFFICIENT MATRICES OF THE STATE
DIFFERENCE EQUATION

MATRIX FM(T)

0.7381E 00	-0.8632E 00	-0.4102E-01
0.8205E-02	0.9971E 00	0.8632E-01
0.1480E 00	-0.8205E-01	0.7381E 00

MATRIX H(T)

0.8632E 00
0.2887E-02
0.8197E-01

TIME DOMAIN RESPONSE

TIME	Y(1)	Y(2)	Y(3)	Y(4)	Y(5)
0.0000	1.0000	0.0000	0.0000	0.0000	0.0000
0.1000	0.9971	0.8632	0.7813	0.0028	0.0819
0.2000	0.9800	1.4946	1.2246	0.0199	0.2700
0.3000	0.9416	1.9383	1.4373	0.0583	0.5009
0.4000	0.8798	2.2231	1.4892	0.1201	0.7338
0.5000	0.7956	2.3704	1.4275	0.2043	0.9429
0.6000	0.6925	2.3979	1.2858	0.3074	1.1121
0.7000	0.5748	2.3223	1.0896	0.4251	1.2326
0.8000	0.4477	2.1599	0.8591	0.5522	1.3007
0.9000	0.3163	1.9275	0.6109	0.6836	1.3165
1.0000	0.1860	1.6419	0.3589	0.8139	1.2830
1.1000	0.0612	1.3200	0.1147	0.9387	1.2053
1.2000	-0.0538	0.9778	-0.1122	1.0538	1.0900
1.3000	-0.1558	0.6306	-0.3142	1.1558	0.9448
1.4000	-0.2421	0.2923	-0.4856	1.2421	0.7779
1.5000	-0.3109	-0.0250	-0.6226	1.3109	0.5975
1.6000	-0.3614	-0.3114	-0.7232	1.3614	0.4117
1.7000	-0.3934	-0.5587	-0.7868	1.3934	0.2281
1.8000	-0.4074	-0.7613	-0.8147	1.4074	0.0533
1.9000	-0.4046	-0.9158	-0.8089	1.4046	-0.1068
2.0000	-0.3867	-1.0208	-0.7731	1.3867	-0.2477
2.1000	-0.3558	-1.0772	-0.7113	1.3558	-0.3658
2.2000	-0.3144	-1.0872	-0.6285	1.3144	-0.4587
2.3000	-0.2650	-1.0551	-0.5296	1.2650	-0.5254
2.4000	-0.2102	-0.9860	-0.4201	1.2102	-0.5659

2.5000	-0.1527	-0.8860	-0.3050	1.1527	-0.5810
2.6000	-0.0948	-0.7619	-0.1893	1.0948	-0.5726
2.7000	-0.0389	-0.6208	-0.0774	1.0389	-0.5433
2.8000	0.0131	-0.4694	0.0267	0.9868	-0.4962
2.9000	0.0598	-0.3147	0.1200	0.9401	-0.4348
3.0000	0.0997	-0.1628	0.1999	0.9002	-0.3627
3.1000	0.1321	-0.0191	0.2646	0.8678	-0.2837
3.2000	0.1563	0.1116	0.3131	0.8436	-0.2015
3.3000	0.1723	0.2257	0.3451	0.8276	-0.1194
3.4000	0.1803	0.3203	0.3610	0.8196	-0.0407
3.5000	0.1806	0.3938	0.3617	0.8193	0.0320
3.6000	0.1741	0.4454	0.3487	0.8258	0.0967
3.7000	0.1616	0.4752	0.3236	0.8383	0.1515
3.8000	0.1441	0.4841	0.2887	0.8558	0.1954
3.9000	0.1229	0.4739	0.2462	0.8770	0.2276
4.0000	0.0990	0.4466	0.1984	0.9009	0.2482
4.1000	0.0736	0.4050	0.1476	0.9263	0.2573
4.2000	0.0478	0.3520	0.0961	0.9521	0.2559
4.3000	0.0227	0.2907	0.0458	0.9772	0.2448
4.4000	-0.0008	0.2242	-0.0013	1.0008	0.2255
4.5000	-0.0221	0.1555	-0.0439	1.0221	0.1995
4.6000	-0.0406	0.0875	-0.0808	1.0406	0.1684
4.7000	-0.0557	0.0226	-0.1111	1.0557	0.1338
4.8000	-0.0673	-0.0368	-0.1343	1.0673	0.0975
4.9000	-0.0753	-0.0893	-0.1502	1.0753	0.0609
5.0000	-0.0796	-0.1333	-0.1588	1.0796	0.0254

```
                                              * JULY 69 *

                  * THIS IS EXAMPLE VI                    *
                  * THIS EXAMPLE STUDIES THE EFFECT OF NE-*
                  * GLIGIBLE TIME CONSTANTS ON THE FINAL  *
                  * RESPONSE                              *

             * INPUT DATA CARDS *

0                           **
0 0 3 50                    **
0 0 1 6 1                   **
0 0 1 8 3                   **
0 0 2 1000. 0.0 0.02 0.002 **

501 2.15 4.0                **
2 0 1.0 0.0 -1 1.0 0.01     **
3 0 1.0 0.0 -1 1. 1. -1 1. 2. **
4 0 1.0 0.0 -1 1.0 0.001    **
101 101 4 1.0 -1.0          **
102 1 1.0                   **
103 2 1.0                   **
104 3 1.0                   **

999 101 1 1.0 0.0           **
0 3 4                       **
```

BLOCK DIAGRAM INPUT

```
INPUT MATRIX FLAG...........................          0
JOB NUMBER..................................          1
PRIORITY NUMBER FOR PRINTING OUTPUT.....              3
LOGICAL UNIT NUMBER FOR INPUT DEVICE....              3
LOGICAL UNIT NUMBER FOR OUTPUT DEVICE...              4

OPTION.. 1..................................          0
OPTION.. 2..................................          0
OPTION.. 3..................................          0
OPTION.. 4..................................          2
OPTION.. 5..................................          0
OPTION.. 6..................................          1
OPTION.. 7..................................          2
OPTION.. 8..................................          3
OPTION.. 9..................................          0
OPTION..10..................................          0

ABSOLUTE CRITERION VALUE (PER CENT).....         0.0000
INITIAL TIME VALUE..........................     0.0000
TIME INTERVAL...............................     0.0200
```

```
TIME SUBINTERVAL..........................   0.0020
NUMBER OF TIME INTERVALS..................       50

INITIAL STATE VARIABLE.. 1................   0.0000
INITIAL STATE VARIABLE.. 2................   0.0000
INITIAL STATE VARIABLE.. 3................   0.0000
INITIAL STATE VARIABLE.. 4................   0.0000
INITIAL STATE VARIABLE.. 5................   0.0000

DESIRED OUTPUT/STATE VARIABLE.. 1.........        1
DESIRED OUTPUT/STATE VARIABLE.. 2.........        2
DESIRED OUTPUT/STATE VARIABLE.. 3.........        3
DESIRED OUTPUT/STATE VARIABLE.. 4.........        4

UTILITY PROGRAM NUMBER....................        0
NUMBER OF JOBS............................        1

DIST.. 1  CODE.. 1  1ST PAR..    1.0000  2ND PAR..    0.0000
```

COEFFICIENT MATRICES OF THE STATE EQUATION
AND RELATED DATA

```
NUMBER OF STATE VARIABLES.................        5
NUMBER OF OUTPUT VARIABLES................        4
NUMBER OF EXTERNAL FORCING FUNCTIONS......        1
NUMBER OF TIME DELAYS.....................        0

NUMBER OF EXTERNAL FORCING FUNCTIONS IN U(T)      1

VARIABLE IN U(T).... 1...DELAY VALUE....   0.0000

NUMBER OF EXTERNAL FORCING FUNCTIONS IN V(T)      1

VARIABLE IN V(T).... 1...DELAY VALUE....   0.0000
```

MATRIX A

0.0000	0.0000	0.0000	0.0000	-0.5375
100.0000	-100.0000	0.0000	0.0000	-214.9999
0.0000	0.0000	0.0000	1.0000	0.0000
0.0000	0.5000	-0.5000	-1.5000	0.0000
0.0000	0.0000	1000.0001	0.0000	-1000.0001

MATRIX B

0.5375

214.9999

0.0000

0.0000

0.0000

MATRIX C

1.0000	0.0000	0.0000	0.0000	-2.1500
0.0000	1.0000	0.0000	0.0000	0.0000
0.0000	0.0000	1.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	1.0000

MATRIX D

2.1500

0.0000

0.0000

0.0000

COEFFICIENT MATRICES OF THE STATE
DIFFERENCE EQUATION

MATRIX FM(T)

0.1000E 01	-0.1621E-06	-0.1021E-01	-0.8847E-04	-0.5375E-03
0.8647E 00	0.1354E 00	-0.1832E 01	-0.2244E-01	-0.3275E-01
0.3538E-04	0.5629E-04	0.1000E 01	0.1973E-01	-0.1122E-04
0.5629E-02	0.4239E-02	-0.2109E-01	0.9704E 00	-0.8994E-03
0.3017E-04	0.5212E-04	0.9998E 00	0.1876E-01	0.2304E-03

MATRIX H(T)

0.1075E-01

0.1871E 01

0.9275E-04

0.1205E-01

0.8156E-04

TIME DOMAIN RESPONSE

TIME	Y(1)	Y(2)	Y(3)	Y(4)
0.0000	2.1500	0.0000	0.0000	0.0000
0.0200	2.1605	1.8713	0.0000	0.0000
0.0400	2.1704	2.1336	0.0005	0.0004
0.0600	2.1794	2.1772	0.0013	0.0013
0.0800	2.1875	2.1903	0.0026	0.0025
0.1000	2.1947	2.1986	0.0042	0.0041
0.1200	2.2011	2.2056	0.0062	0.0061
0.1400	2.2067	2.2116	0.0086	0.0085
0.1600	2.2115	2.2167	0.0114	0.0112
0.1800	2.2155	2.2211	0.0145	0.0143
0.2000	2.2187	2.2247	0.0179	0.0178
0.2200	2.2211	2.2275	0.0217	0.0215
0.2400	2.2228	2.2296	0.0259	0.0256
0.2600	2.2238	2.2310	0.0303	0.0301
0.2800	2.2241	2.2316	0.0350	0.0348
0.3000	2.2237	2.2316	0.0400	0.0398
0.3200	2.2227	2.2309	0.0454	0.0451
0.3400	2.2210	2.2295	0.0509	0.0507
0.3600	2.2186	2.2275	0.0568	0.0565
0.3800	2.2157	2.2248	0.0629	0.0626
0.4000	2.2121	2.2216	0.0693	0.0689
0.4200	2.2080	2.2177	0.0759	0.0755
0.4400	2.2033	2.2133	0.0827	0.0824
0.4600	2.1981	2.2083	0.0897	0.0894
0.4800	2.1923	2.2028	0.0970	0.0966
0.5000	2.1860	2.1968	0.1045	0.1041
0.5200	2.1792	2.1902	0.1121	0.1118
0.5400	2.1719	2.1832	0.1200	0.1196
0.5600	2.1642	2.1757	0.1280	0.1276
0.5800	2.1560	2.1677	0.1362	0.1358
0.6000	2.1473	2.1593	0.1446	0.1442
0.6200	2.1383	2.1505	0.1531	0.1527
0.6400	2.1288	2.1412	0.1618	0.1613
0.6600	2.1190	2.1315	0.1706	0.1701
0.6800	2.1088	2.1215	0.1795	0.1790
0.7000	2.0982	2.1111	0.1885	0.1881
0.7200	2.0872	2.1003	0.1977	0.1973
0.7400	2.0760	2.0892	0.2070	0.2065
0.7600	2.0644	2.0778	0.2164	0.2159
0.7800	2.0525	2.0661	0.2259	0.2254
0.8000	2.0403	2.0540	0.2354	0.2349

0.8200	2.0279	2.0417	0.2451	0.2446
0.8400	2.0151	2.0291	0.2548	0.2543
0.8600	2.0022	2.0163	0.2646	0.2641
0.8800	1.9890	2.0032	0.2744	0.2739
0.9000	1.9755	1.9899	0.2843	0.2839
0.9200	1.9619	1.9763	0.2943	0.2938
0.9400	1.9481	1.9626	0.3043	0.3038
0.9600	1.9340	1.9487	0.3143	0.3138
0.9800	1.9198	1.9345	0.3244	0.3239
1.0000	1.9055	1.9202	0.3345	0.3340

```
                                                          * JULY 69 *

                  * THIS IS EXAMPLE VII                                *
                  * THIS EXAMPLE SHOWS THE EFFECT OF TIME              *
                  * INTERVAL ON THE TIME DOMAIN RESPONSE               *
                  * FOR CAYLEY-HAMILTON TECHNIQUE AND POWER            *
                  * SERIES METHOD.RANGE OF DELT IVESTIGATED            *
                  * IS 0.001 (RIGHT CHOICE) TO 0.1                     *

        * INPUT DATA CARDS *

0                             **
0 0 5 3                       **
0 0 3 1000 **
0 0 2 100. 0.0 0.005 0.0005 **
0 0 1 6 1                     **
0 0 1 8 3                     **

501 2.15 4.0                  **
2 0 1.0 0.0 -1 1.0 0.01       **
3 0 1.0 0.0 -1 1. 1. -1 1. 2.0   **
4 0 1.0 0.0 -1 1.0 0.001      **
101 101 4 1.0 -1.0            **
102 1 1.0                     **
103 2 1.0                     **
104 3 1.0                     **

999 101 1 1.0 0.0             **
0 3 4                         **


                      BLOCK DIAGRAM INPUT


INPUT MATRIX FLAG........................            0
JOB NUMBER...............................            1
PRIORITY NUMBER FOR PRINTING OUTPUT.....             3
LOGICAL UNIT NUMBER FOR INPUT DEVICE....             3
LOGICAL UNIT NUMBER FOR OUTPUT DEVICE...             4

OPTION.. 1...............................            0
OPTION.. 2...............................            0
OPTION.. 3...............................            0
OPTION.. 4...............................            2
OPTION.. 5...............................            0
OPTION.. 6...............................            1
OPTION.. 7...............................            2
OPTION.. 8...............................            3
OPTION.. 9...............................            0
OPTION..10...............................            0
```

```
ABSOLUTE CRITERION VALUE (PER CENT).....     0.0000
INITIAL TIME VALUE......................     0.0000
TIME INTERVAL...........................     0.0050
TIME SUBINTERVAL........................     0.0005
NUMBER OF TIME INTERVALS................       1000

INITIAL STATE VARIABLE.. 1..............     0.0000
INITIAL STATE VARIABLE.. 2..............     0.0000
INITIAL STATE VARIABLE.. 3..............     0.0000
INITIAL STATE VARIABLE.. 4..............     0.0000
INITIAL STATE VARIABLE.. 5..............     0.0000

DESIRED OUTPUT/STATE VARIABLE.. 1.......          3

UTILITY PROGRAM NUMBER..................          0
NUMBER OF JOBS..........................          1

DIST.. 1  CODE.. 1  1ST PAR..    1.0000  2ND PAR..    0.0000
```

COEFFICIENT MATRICES OF THE STATE EQUATION
AND RELATED DATA

```
NUMBER OF STATE VARIABLES...............          5
NUMBER OF OUTPUT VARIABLES..............          4
NUMBER OF EXTERNAL FORCING FUNCTIONS....          1
NUMBER OF TIME DELAYS...................          0

NUMBER OF EXTERNAL FORCING FUNCTIONS IN U(T)      1

VARIABLE IN U(T).... 1...DELAY VALUE....     0.0000

NUMBER OF EXTERNAL FORCING FUNCTIONS IN V(T)      1

VARIABLE IN V(T).... 1...DELAY VALUE....     0.0000
```

MATRIX A

0.0000	0.0000	0.0000	0.0000	-0.5375
100.0000	-100.0000	0.0000	0.0000	-214.9999
0.0000	0.0000	0.0000	1.0000	0.0000
0.0000	0.5000	-0.5000	-1.5000	0.0000
0.0000	0.0000	1000.0001	0.0000	-1000.0001

MATRIX B

0.5375

214.9999

0.0000

0.0000

0.0000

MATRIX C

1.0000	0.0000	0.0000	0.0000	-2.1500
0.0000	1.0000	0.0000	0.0000	0.0000
0.0000	0.0000	1.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	1.0000

MATRIX D

2.1500

0.0000

0.0000

0.0000

COEFFICIENT MATRICES OF THE STATE
DIFFERENCE EQUATION

MATRIX FM(T)

0.1000E 01	-0.2809E-08	-0.2153E-02	-0.4521E-05	-0.5338E-03
0.3934E 00	0.6065E 00	-0.7030E 00	-0.1584E-02	-0.1434E 00
0.8909E-06	0.5311E-05	0.9999E 00	0.4981E-02	-0.7921E-06
0.5311E-03	0.1959E-02	-0.3282E-02	0.9925E 00	-0.3503E-03
0.5226E-06	0.3683E-05	0.9932E 00	0.3993E-02	0.6738E-02

MATRIX H(T)

0.2687E-02

0.8467E 00

0.1990E-05

0.1141E-02

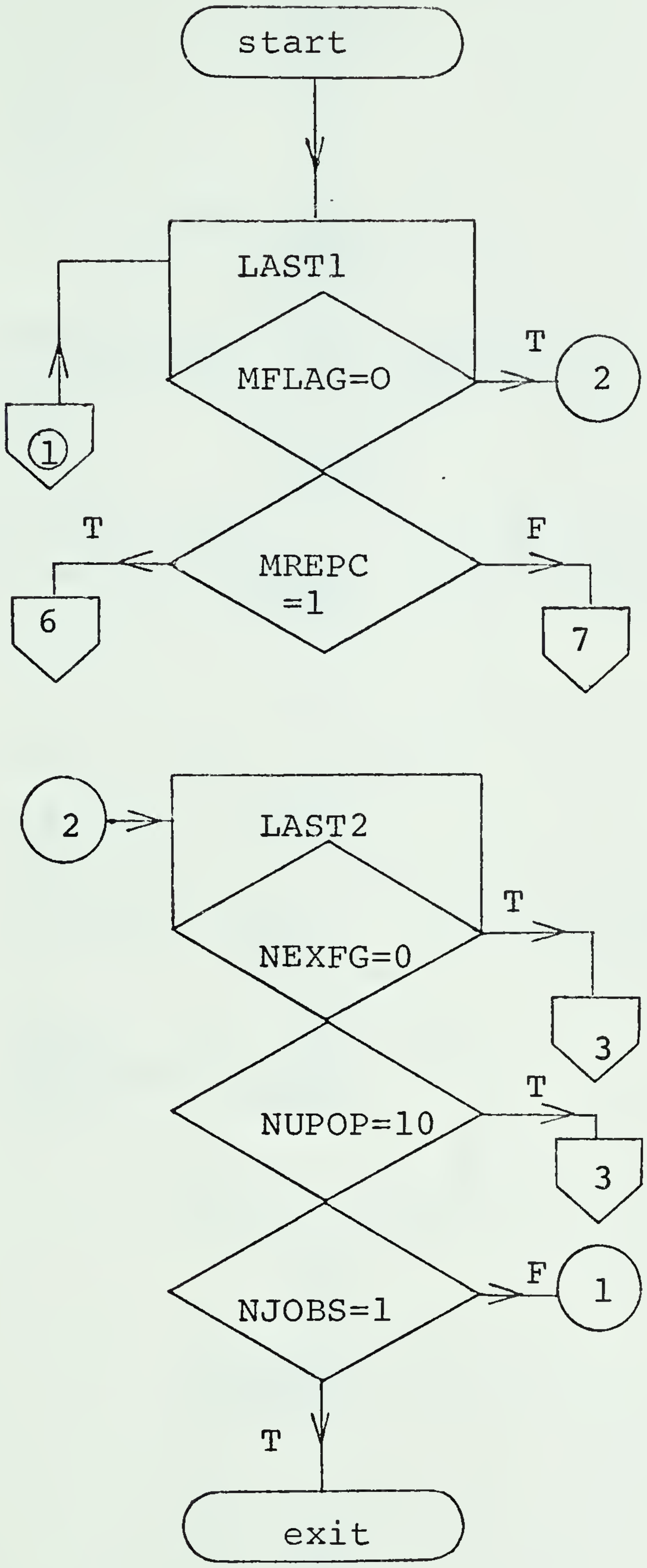
0.1196E-05

TIME DOMAIN RESPONSE

TIME	Y(3)
0.0000	0.0000
0.0050	0.0000
0.0100	0.0000
0.0150	0.0000
0.0200	0.0000
0.0250	0.0001
0.0300	0.0002
0.0350	0.0003
0.0400	0.0005
0.0450	0.0006
0.0500	0.0008
0.0550	0.0011
0.0600	0.0013
0.0650	0.0016
0.0700	0.0019
0.0750	0.0022
0.0800	0.0026
0.0850	0.0029
0.0900	0.0033
0.0950	0.0037
0.1000	0.0042
0.1050	0.0047
0.1100	0.0052
0.1150	0.0057
0.1200	0.0062
0.1250	0.0068
0.1300	0.0074
0.1350	0.0080
0.1400	0.0086
0.1450	0.0092
0.1500	0.0099
0.1550	0.0106
0.1600	0.0113
0.1650	0.0121
0.1700	0.0129
0.1750	0.0136
0.1800	0.0144
0.1850	0.0153
0.1900	0.0161
0.1950	0.0170
0.2000	0.0179

APPENDIX D

System Documentation



comments:

LAST1 provides all data for a TF+configuration input

MFLAG=matrix input flag

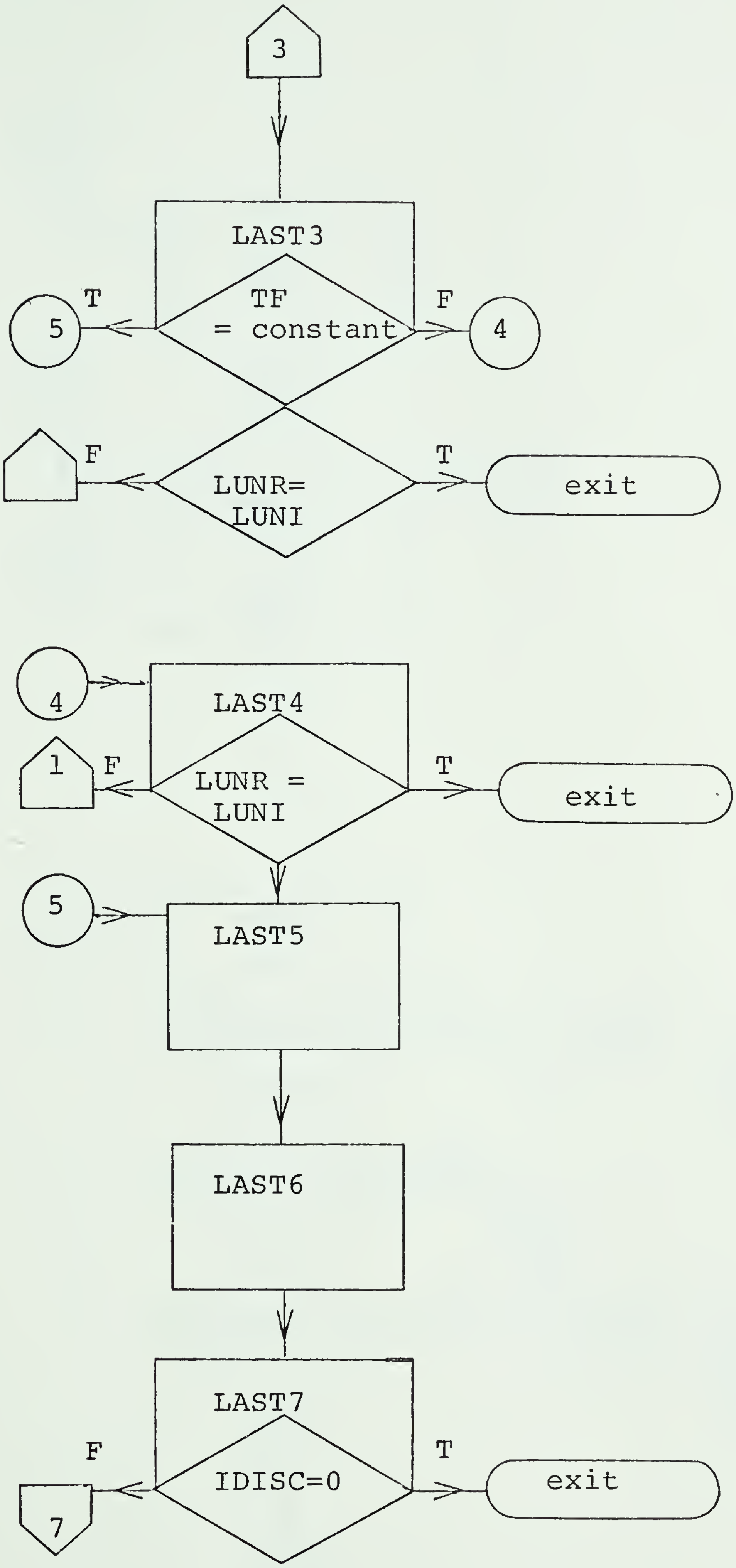
MREPC=repetition counter (=1 means first time through etc..)

LAST2 calculates
N=number of state variables
M=number of ouput variables
NF=number of external forcing functions

NEXFG=non execution flag
NUPOP=utility program option
NJOBS=number of jobs

FIGURE D-1 Flow Diagram by Links

...continue



LAST3 sorts input data and provides controller constants
TF=transfer function

If the I/O device is 1816 keyboard program allows entry of

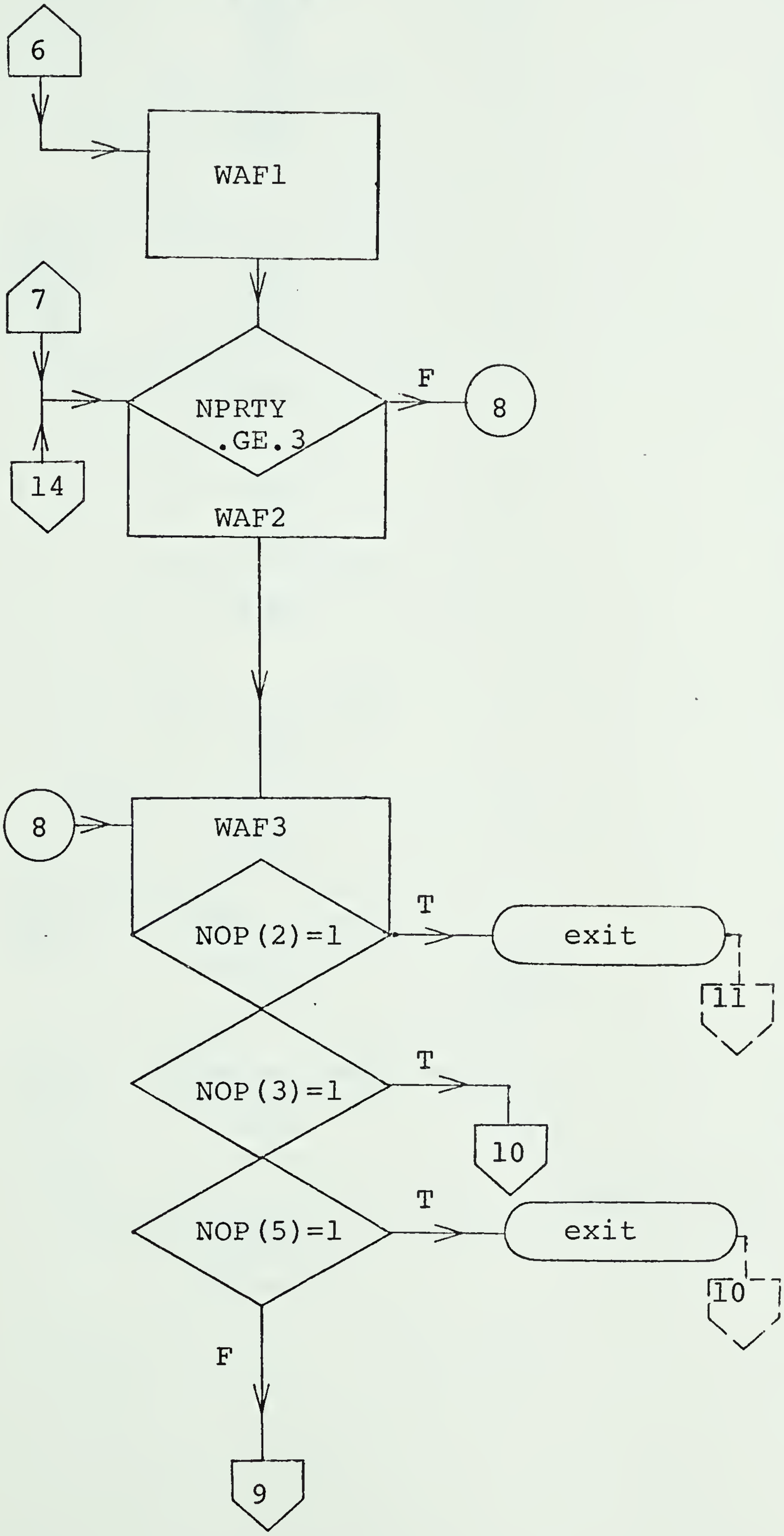
LAST4 finds the coefficients of the state equation

LAST5 sets coefficients of state equations into matrices

LAST6 manipulates matrices to yield the final $\underline{A}$, $\underline{B}$, $\underline{C}$, $\underline{D}$

LAST7 manipulates matrices in special cases to yield the final $\underline{A}$, $\underline{B}$, $\underline{C}$, $\underline{D}$,
IDISC= disturbance counter

...continue



WAF1 completes the input of data for the case of "matrix input"

WAF2 provides a record of input data if NPRTY is greater or equal 3

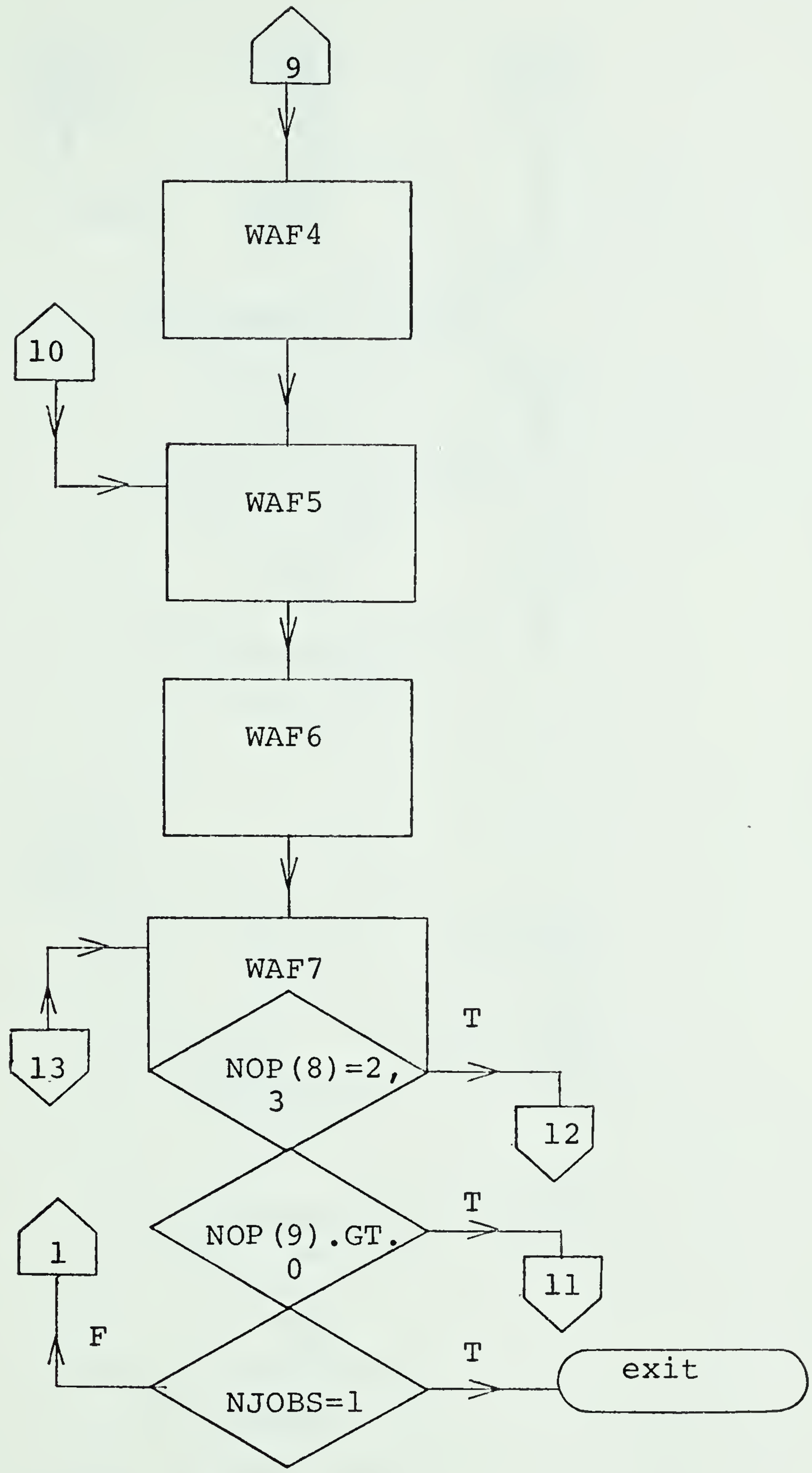
WAF3 provides the eigenvalues of coefficient matrix $\underline{A}$

NOP(2)=1: direct integration of state equations by Runga-Kutta (not implemented)

NOP(3)=1: power series method to calculate the fundamental matrix $\underline{\emptyset}$

NOP(5)=1: similarity transformation to calculate $\underline{\emptyset}$ (not implemented)

...continue



WAF4 checks the eigenvalues and builds the matrix <u>CC</u> according to the Cayley-Hamilton technique

WAF5 calculates the coefficient matrices of the state difference equation $\underline{\emptyset}$(T) and H(T) over n-1 time subintervals

WAF6 completes the calculation of $\underline{\emptyset}$(T),$\underline{H}$(T)

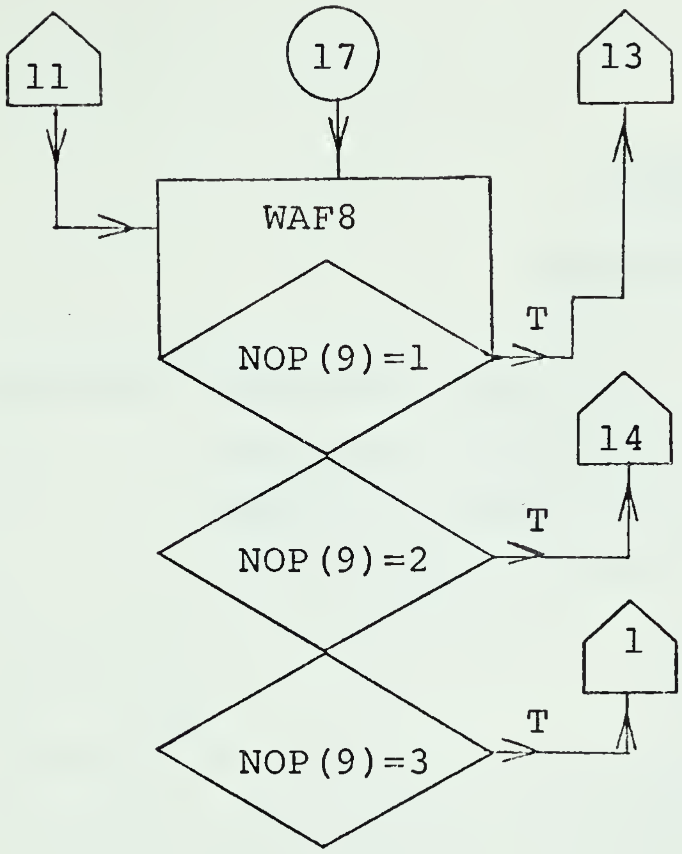
WAF7 find the time domain response

NOP(8)=2,3 provide a plot on the scope or plotter

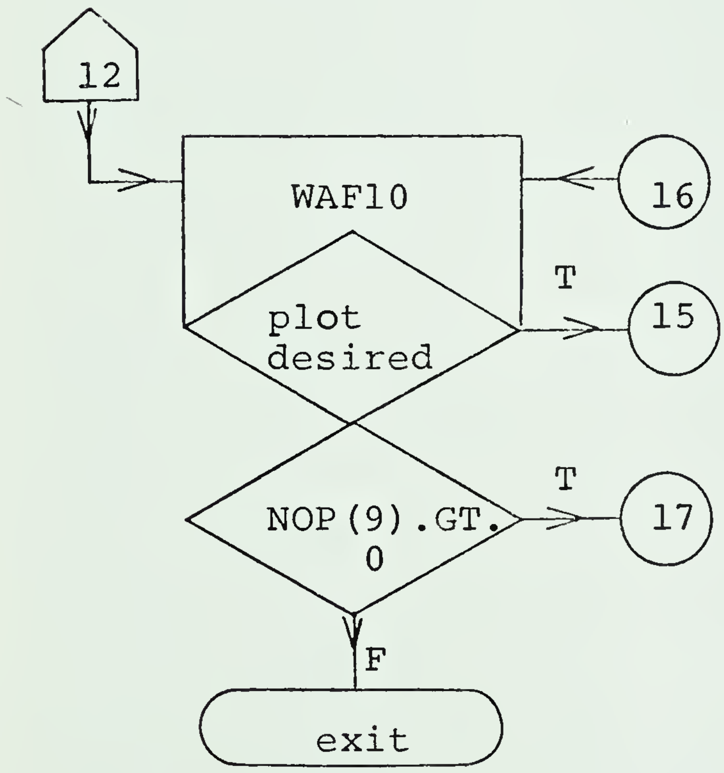
NOP(9) greater than zero implies that some change is desired

NJOBS=number of jobs to be executed

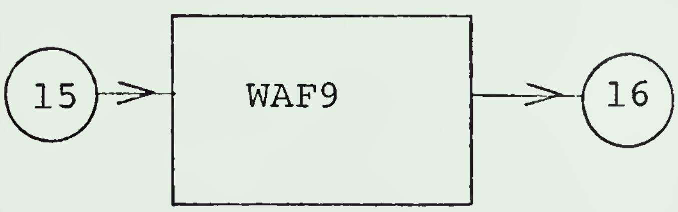
...continue



WAF8 allows the user to make changes and rerun the problem



WAF10 supplies data to the plotting routine



WAF9 plots variables of interest on the scope or on the plotter

TABLE D-1

FORTRAN COMMON (*)

```
DIMENSION IDUM1(80),RDUM1(45),IDUM2(290),RDUM2(330),IDUM3(70),
          RDUM3(80),RDUM4(300),DUM1(300),DUM2(225),DUM3(225),
          MBLOK(50),RBLOK(40),RTEMP(300),IW(10),RW(40),FM(100),
          H(200),IVECT(20),LABX(10),LABY(10),IPTNY(10),IDUM9(21)
```

Link	COMMON
LAST1	IDUM1,RDUM1,IDUM2,RDUM2
LAST2	IDUM1,RDUM1,IDUM2,RDUM2
LAST3	IDUM1,RDUM1,IDUM2,RDUM2,IDUM3,RDUM3
LAST4	IDUM1,RDUM1,IDUM2,RDUM2,IDUM3,RDUM3
LAST5	IDUM1,RDUM1,IDUM2,RDUM2,IDUM3,RDUM3,RDUM4
LAST6	IDUM1,RDUM1,MBLOK,RBLOK,DUM1,DUM2,DUM3
LAST7	IDUM1,RDUM1,MBLOK,RBLOK,DUM1,DUM2,DUM3
WAF1	IDUM1,RDUM1,MBLOK,RBLOK,RTEMP
WAF2	IDUM1,RDUM1,MBLOK,RBLOK,RTEMP
WAF3	IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW
WAF4	IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW
WAF5	IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW,FM,H
WAF6	IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW,FM,H
WAF7	IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW,FM,H
WAF8	IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW,FM,H
WAF10	IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW,FM,H
WAF9	IVECT,LABX,LABY,IPTNY,IRET,IPTAL,IPTCT,KY,ID,IG, IU,IP,KN,IDUM9,XMAX,XMIN,YMAX,YMIN

(*) One word integers,single precision

TABLE D-2

Disk Files

File Definition	Number of Sectors	File Content
40(95,10,U,NEXT)	3	input data for reexecution
51(100,2,U,NXT1)	1	matrix A1
52(150,2,U,NXT2)	1	B1
53(100,2,U,NXT3)	1	C1
54(150,2,U,NXT4)	1	F
55(225,2,U,NXT5)	2	G
56(150,2,U,NXT6)	1	H
57(225,2,U,NXT7)	2	AL
58(150,2,U,NXT8)	1	AM
59(225,2,U,NXT9)	2	AL1
60(150,2,U,NXT10)	1	D1
61(100,2,U,NXT11)	1	E1
62(150,2,U,NXT12)	1	D2
63(100,2,U,NXT13)	1	E2
70(100,2,U,NXT70)	1	AT
71(100,2,U,NXT71)	1	BT
72(150,2,U,NXT72)	1	CT
73(150,2,U,NXT73)	1	DT
74(225,2,U,NXT74)	2	AL2
75(150,2,U,NXT75)	1	AM2
76(225,2,U,NXT76)	2	Q2
77(225,2,U,NXT77)	2	A2
78(225,2,U,NXT78)	2	C2
79(150,2,U,NXT79)	1	B2
80(150,2,U,NXT80)	1	B3
81(150,2,U,NXT81)	1	B4

...continue

82(100,2,U,NXT82)	1	<u>B5</u>
83(100,2,U,NXT83)	1	<u>WA1</u>
84(100,2,U,NXT84)	1	<u>WA2</u>
85(100,2,U,NXT85)	1	<u>WA3</u>
91(100,2,U,NXT91)	1	<u>A</u>
92(200,2,U,NXT92)	2	<u>B</u>
93(150,2,U,NXT93)	1	<u>C</u>
94(300,2,U,NXT94)	2	<u>D</u>
96(1120,2,U,NXT96)	7	time domain response data
99(160,10,U,NXT99)	5	COMMON of Link WAF10
100(1120,2,U,NX100)	7	desired output variable data
101(1120,2,U,NX101)	7	independent variable (time)

TABLE D-3

Use of Files

Link	File Number	Link	File Number	Link	File Number
LAST1	40		92		78
LAST2	40		93		79
LAST3	--		94		80
LAST4	--		96		81
LAST5	40		99		82
	91		51		83
	92		52		84
	93		53		85
	94		54	WAF1 &	40
	96		55	WAF2	91
	99		56		92
	51		57		93
	52		58		94
	53		59	WAF3	--
	54		60	WAF4	--
	55		61	WAF5	--
	56		62	WAF6	40
	57		63		91
	58		70		92
	59		71	WAF7	40
	60		72		91
	61		73		92
	62		74		93
	63		75		94
LAST6 &	40		76		96
LAST7	91		77	WAF8	--

...continue

WAF9	40
	91
	92
	93
	94
	96
	99
	100
	101
WAF10	40
	91
	92
	93
	94
	96
	99
	100

APPENDIX E

Listing of Program

```
      C
C
C                                 THIS IS LINK 1 OF PART A
C      THE FOLLOWING TWO FLAGS MUST BE IN COMMON TO PROVIDE
C      REENTRY AT
C      DIFFERENT LEVELS.
C      IFLG1 IS SET IF AN ERROR OCCURS DURING EXECUTION OF
C      NEXT LINK.
C      IFLG2 IS SET AT THE END OF THE PROGRAM TO MAKE ANY
C      SORT OF CHANGE
C      WHEN THIS IS DONE  INPUT FROM DISK FLAG MUST BE SET TO
C      1 (IFDFG) A
C      THE WHOLE COMMON AREA IS DESTROYED .
C
C

       EXTERNAL LAST2,WAF1,WAF3

C

       DIMENSION IHV(20),NBLKV(30),ITFV(30),ICDV(30),NIV(30)
      1,NRV(30),
      1IV(120),RV(330)
       DIMENSION IW(25),RW(50),IALPH(100)
       DIMENSION NY(15),NCODE(10),VD1(10),VD2(10),X(10)
      1,NOP(10)
       DIMENSION IDUM1(80)
       DIMENSION RDUM1(45)
       DIMENSION IDUM2(290),RDUM2(330)

       COMMON IDUM1,RDUM1,IDUM2,RDUM2

       EQUIVALENCE   (IDUM1(1),LUNR),(IDUM1(2),LUNP)
      1,(IDUM1(3),LUNI)
       EQUIVALENCE (IDUM1(4),LUNO),(IDUM1(6),M),(IDUM1(12)
      1,N5)
       EQUIVALENCE (IDUM1(14),NPTS),(IDUM1(15),NPRTY)
       EQUIVALENCE (IDUM1(16),IFLG2),(IDUM1(17),IFDFG)
      1,(IDUM1(18),JOBNO)
       EQUIVALENCE (IDUM1(19),NUPOP),(IDUM1(20),INO)
      1,(IDUM1(21),IDISC)
       EQUIVALENCE (IDUM1(22),ISECT),(IDUM1(23),IFLG1)
      1,(IDUM1(29),MMAX)
       EQUIVALENCE (IDUM1(30),MAXOP),(IDUM1(31),MAXX)
      1,(IDUM1(32),M1)
       EQUIVALENCE (IDUM1(33),IHMAX),(IDUM1(34),IVMAX)
      1,(IDUM1(35),IRVMX)
       EQUIVALENCE (IDUM1(36),INYFG),(IDUM1(38),IVCT)
       EQUIVALENCE (IDUM1(39),IRVCT),(IDUM1(40),IREPC)
      1,(IDUM1(41),NJOBS)
```

```
      C
       EQUIVALENCE (IDUM1(42),NEXFG),(IDUM1(43),MREPC)
       EQUIVALENCE (IDUM1(46),NY(1)),(IDUM1(61),NCODE(1))
      1,(IDUM1(71),NOP
      11))

       EQUIVALENCE (RDUM1(1),CRIT),(RDUM1(2),TO),(RDUM1(3)
      1,DELT)
       EQUIVALENCE (RDUM1(4),DELTH)
       EQUIVALENCE (RDUM1(11),VD1(1))
       EQUIVALENCE (RDUM1(21),VD2(1)),(RDUM1(31),X(1))

       EQUIVALENCE (IDUM2(1),IHV(1)),(IDUM2(21),NBLKV(1))
      1,(IDUM2(51),ITF
      1(1)),(IDUM2(81),ICDV(1)),(IDUM2(111),NIV(1))
      1,(IDUM2(141),NRV(1)),
      2(IDUM2(171),IV(1))

       EQUIVALENCE (RDUM2(1),RV(1))
C
C      ONLY ONE JOB CAN BE STORED ON DISK.IF MORE THAN ONE
C      ARE TO BE STOR
C      THEN EXTRA FILES MUST BE DEFINED ALONG WITH THE
C      FOLLOWINF VECTORS$
C      IHV,NBLKV,ITFV,ICDV,NIV,NRV,IV,RV.THIS,HOWEVER, IS A
C      HIGHLY UNLI-
C      KELY SITUATION.
C      SEVERAL JOBS CAN BE EXECUTED SEQUENTIALLY BY USING THE
C      SAME DISK
C      FILE AND ENTERING THE NUMBER OF JOBS AS SPECIAL ENTRY
C      0 0 NJOBS **
C
C

       DEFINE FILE 40(95,10,U,NEXT)

C

       IF(IFLG1-99) 1,30,1
     1 IF(IFLG2-99) 2,18,2
     2 CONTINUE

       JOBNO=1
       NPRTY=3
       LUNI=5
       LUNO=6

C      ABOVE VALUES ARE THE STANDARD ONES.IF ANY CHANGE IN
C      ANY OF THEM IS
C      DESIRED THE 0 CODE MAY BE USED AS   0,JOB NUMBER,NPRTY
```

```
      C
C     ,LUNI,LUNO *
C     IN PARTICULAR IF ONLY LOGICAL UNIT NUMBERS ARE TO BE
C     CHANGED USE T
C      O CODE AS  O,LUNI,LUNO **.THE FOLLOWING FFIOR USES
C      LUNI AND LUNO
C     AS DEFINED ABOVE.
C     NOTE THAT THE JOB NUMBER MUST BE A NON ZERO INTEGER.
C     A SECOND AND THIRD LOGIC UNIT NUMBERS ARE EQUATED TO
C     THOSE OF THE
C     PRINTER AND READER AND WILL NOT BE CHANGED.
C

      LUNP=6
      LUNR=5

      WRITE(LUNP,4)
    4 FORMAT(1H1)

      NR=-1
      CALL FFIOR(IW,RW,IALPH,NI,NR,NA,LUNO,LUNI)

      MFLAG=IW(1)
      NOP(1)=MFLAG

C     MAXOP IS THE MAXIMUM NUMBER OF OPTIONS SET TO 10.
C     MAXX IS THE MAXIMUM NUMBER OF STATE VARIABLES SET TO
C     10.
C     MMAX IS THE MAXIMUM NUMBER OF OUTPUT VARIABLES SET TO
C     15.
C

      MMAX=15
      MAXOP=10
      MAXX=10
      M1=2*MMAX
      IF(MFLAG) 6,6,10
    6 CONTINUE

C     THE FOLLOWING CARDS DEFINE THE MAXIMUM DIMENSIONS  OF
C     VECTORS
C     IHV,NBLKV,ITFV,ICDV,NIV,NRV,IV,RV AND ARE THE ONLY
C     ONES THAT NEED
C     TO BE CHANGED IF THE PROBLEM IS SCALED UP.LENGTHS OF
C     IV AND RV ARE
C     ESTIMATED ON AVERAGE DIMENSIONS OF TRANSFER FUNCTIONS
C     AND ITS I/O
C     PARAMETERS.
C

      IHMAX=20
```

```
C
      IVMAX=120
      IRVMX=330
C     ZERO THE FOLLOWING VECTORS IHV,NBLKV,ITFV,ICDV,NIV,NRV
C     ,IV
C
      DO 7 I=1,IHMAX
      IHV(I)=0
    7 CONTINUE
      DO 8 I=1,M1
      NBLKV(I)=0
      ITFV(I)=0
      ICDV(I)=0
      NIV(I)=0
      NRV(I)=0
    8 CONTINUE
      DO 9 I=1,IVMAX
      IV(I)=0
    9 CONTINUE
C     THIS PART INITIALIZES ALL SPECIAL ENTRIES IN ORDER TO
C     PROVIDE
C     A '' STANDARD'' SOLUTION OF THE PROBLEM.(SEE TABLE OF
C     OPTIONS).
C
C     INITIALIZE VECTOR OF OPTION.
C
   10 CONTINUE
      DO 13 I=2,MAXOP
      NOP(I)=0
   13 CONTINUE
      NOP(4)=2
      NOP(7)=2
      NOP(8)=1
C     INITIALIZE CRIT,TO,DELT,DELTH,NPTS.
C
      CRIT=3.
      TO=0.0
      DELT=0.1
      DELTH=0.01
      NPTS=10
C     INITIALIZE STATE VARIABLE VECTOR.
C
      DO 14 I=1,MAXX
```

```
      C
      X(I)=0.0
   14 CONTINUE
C     SET UTILITY PROGRAM OPTION TO ZERO.
C
      NUPOP=0
C     SET NUMBER OF JOBS EQUAL TO ONE.
C
      NJOBS=1
C     SET REPETITION COUNTER FOR MATRIX INPUT TO ZERO.
C
      MREPC=0
C     ZERO DISTURBANCE INPUT COUNTER.IF  IDISC IS GREATER
C     THAN 1 THE PRO
C     ASSUMES THAT THE USER WHISHES TO CONTINUE TO PART B OF
C     THE PROGRAM
C     AS THIS TYPE OF INPUT IS ONLY NEEDED IN PART B.
C
      IDISC=0
C     SET FLAG FOR SPECIAL NY ENTRIES TO ZERO.
C
      INYFG=0
      IF(MFLAG) 15,15,35
   15 CONTINUE
C     ZERO INPUT FROM DISK FLAG
C
      IFDFG=0
C     ZERO OUTPUT VARIABLE COUNTER.
C
      M=0
C     SET INITIAL VALUE COUNTERS OUTSIDE THE LOOP AROUNF
C     FFINP AND AHEAD
C     OF THE REENTRY STATEMENTS.
C
      INO=1
```

```
      C
      IVCT=0
      IRVCT=0
      IREPC=0
   18 CONTINUE
C     RESET REENTRY FLAG TO ZERO.
C
      IFLG2=0
C     DECIDE WHETHER DATA MUST BE READ FROM DISK OR IS
C     INPUTTED NOW.
C
      IF(IFDFG)31,31,20
   20 K=1
      READ(40'K)IHV,NBLKV,ITFV,ICDV,NIV,NRV,IV,RV
C     NEXT OPERATOR ENTRY WILL BE 999,0 **.
C
   30 CONTINUE
C      RESET REENTRY FLAG TO ZERO
C
      IFLG1=0
   31 CONTINUE
   35 NR=-1
      CALL FFIOR(IW,RW,IALPH,NI,NR,NA,LUNO,LUNI)
C     START ANALYZING THE TYPE OF INPUT.
C
      NBK=IW(1)
      IF(NBK) 50,50,40
   40 IF(NBK-999) 60,300,300
C     FIND OUT IF IT IS A SPECIAL INPUT. CODE NUMBERS ARE 0
C     ,0    .
C
   50 IF(IW(2)) 400,400,51
C     IT IS A '0' INPUT.
C
   51 CONTINUE
```

```
      C
      IF(NI-3) 55,55,52
C     IT IS THE FIRST '0' INPUT.
C     STORE INPUT DATA IN PROPER LOCATION.FIRST THE JOB
C     NUMBER.
C
   52 IHV(1)=IW(2)
      JOBNO=IW(2)
C     ADD INTO IHV VECTOR THE FOLLOWING DATA  TO BE
C     ASSOCIATED WITH
C     ITS DISK FILE $ FILE NO.,SECTOR COUNT,8 ADDRESSES FOR
C     THE FOLLOWIN
C     VECTORS $ IHV,NBLKV,ITFV,ICDV,NIV,NRV,IV,RV.
C
      IHV(2)=40
      IHV(3) =3
      IHV(4)=1
      IHV(5)=21
      IHV(6)=51
      IHV(7)=81
      IHV(8)=111
      IHV(9)=141
      IHV(10)=171
      IHV(11)=291
C     ABOVE DATA TO BE REPEATED FOR EACH SEPARATE DISK FILE.
C
      NPRTY=IW(3)
      LUNI=IW(4)
      LUNO=IW(5)
      GO TO 35
C     IT IS A SECOND OR SUCCESSIVE '0' ENTRY.
C
   55 LUNI=IW(2)
      LUNO=IW(3)
      GO TO 35
   60 CONTINUE
C     TO DECIDE WHETHER IT IS A NEW ENTRY OR A CORRECTION OF
C     AN EXISTING
C     ONE THE TRANSFER FUNCTION ID 'S MUST BE COMPARED
C     FOR EXAMPLE 3,203,603 ... ALL MEAN THE SAME TF G(3).
C
```

```
      C

      NCD=0
      NB=NBK
   61 IF(NB-100) 65,65,62
   62 NCD=NCD+1
      NB=NB-100
      GO TO 61

C     NB IS THE INPUT ID.
C     NCD IS THE INPUT CODE NUMBER.
C     JUMP THE COMPARISON THE FIRST TIME THROUGH
C

   65 IF(INO-1) 150,150,70

C     DISTINGUISH BETWEEN A TRANSFER FUNCTION AND AN I/O
C     DATA SET
C

   70 INO1=INO-1
      IF(NCD-1) 71,75,71

C     IT IS A T.F.
C

   71 DO 73 J=1,INO1
      IF(ITFV(J)-NB) 73,81,73
   73 CONTINUE

C     RESET REPETITION COUNTER TO ZERO.
C

      IREPC=0
      GO TO 140

C     IT IS AN I/O DATA SET
C

   75 DO 76 J=1,INO1
      IF(NBLKV(J)-NBK) 76,81,76
   76 CONTINUE

C     IT IS A NEW I/O ENTRY.
C     CHECK FOR SPACE IN NBLKV VECTOR.
C

      IF(INO-M1) 80,80,77
   77 WRITE(LUNO,78)
   78 FORMAT(1H ,'TOO MANY VALUES IN VECTOR NBKLV')
      CALL EXIT
   80 NBLKV(INO)=NBK
```

```
      C
      GO TO 190
C     IT IS NOT A NEW T.F. ENTRY.
C     CHECK IF BOTH NUMBER OF INTEGERS AND REAL VALUES
C     COINCIDE.
C
   81 IF(NCD-1) 810,810,808
  808 IF(NIV(J)-NI) 100,82,100
  810 IF(NIV(J)-NI+1) 100,82,100
   82 IF(NRV(J)-NR) 100,83,100
C     FIND ADDRESSES OF OLD ENTRIES IN IV AND RV.
C
   83 ITEM=0
      ITEM1=0
      J1=J-1
C     JUMP THE SUMMING STATION IF J IS ONE.
C
      IF(J1) 88,88,85
   85 DO 86 KK=1,J1
      ITEM=ITEM+NIV(KK)
      ITEM1=ITEM1+NRV(KK)
   86 CONTINUE
C     BYPASS THIS ENTRY IF NI=1.
C
   88 IF(NI-1) 93,93,90
   90 KK1=NI-1
C     ENTER NEW VALUES ON TOP OF THE OLD ONES.
C
      DO 91 KK=1,KK1
      J1=ITEM+KK
      IV(J1)=IW(KK+1)
   91 CONTINUE
C     BYPASS THIS ENTRY IF NR IS ZERO.
C
   93 IF(NR) 99,99,95
   95 DO 96 KK=1,NR
      J1=ITEM1+KK
      RV(J1)=RW(KK)
   96 CONTINUE
```

```
      C

C     WRITE CODE NO. AND BLOCK NO. ON TOP OF THE OLD ONES.
C

   99 NBLKV(J)=NBK
      ICDV(J)=NCD
      GO TO 35
C     STEP REPETITION COUNTER.
C

  100 IREPC=IREPC+1
C     ZERO OLD ENTRIES.
C

      NBLKV(J)=0
      ITFV(J)=0
      ICDV(J)=0
C     ZERO OLD ENTRIES IN IV VECTOR
C

      IF(J-1) 120,120,130
  120 IV(J)=0
      GO TO 140
  130 ISUM=0
      J1=J-1
      DO 132 I=1,J1
      ISUM=ISUM+NIV(I)
  132 CONTINUE
      J1=ISUM+NIV(J)
      ISUM=ISUM+1
      DO 135 I=ISUM,J1
      IV(I)=0
  135 CONTINUE
C     IT IS A NEW T.F.
C

  140 IF(INO-M1) 150,150,77
  150 NBLKV(INO)=NBK
      IF(NCD-1) 155,190,155
  155 ITFV(INO)=NB
      ICDV(INO)=NCD
      IF(NCD) 180,180,160
C     THESE ARE SPECIAL ENTRIES INTO IV VECTOR TO ALLOW N TO
C     BE CORRECTL
C     CALCULATED.
```

```
      C
C
  160 GO TO (190,165,170,175,165,170,176,176),NCD

  165 NI=NI+1
      IW(NI)=-1
      GO TO 180
  170 NI=NI+1
      IW(NI)=-2
      GO TO 180
  175 NI=NI+1
      IW(NI)=0
      GO TO 180

  176 WRITE(LUNO,177) NB
  177 FORMAT(1H ,'ILLEGAL CODE NUMBER IN T.F.',I5/)
  180 IF(IREPC) 185,185,190
  185 M=M+1
  190 CONTINUE

C     STORE NI,NR,IW,RW VALUES IN THEIR RESPECTIVE VECTORS.
C

      NIV(INO)=NI-1
      NRV(INO)=NR

C     BYPASS THIS ENTRY IF NI=1.
C

      IF(NI-1) 215,215,199

C     FIND UPPER POSITION OF NEW ENTRY AND CHECK IT AGAINST
C     IVMAX,IRVMX.
C

  199 J1=IVCT+NI-1
      IF(J1-IVMAX) 205,205,200
  200 WRITE(LUNO,201)
  201 FORMAT(1H ,'TOO MANY VALUES IN IV VECTOR'/)
      CALL EXIT
  205 IVCT1=IVCT+1
      DO 210 J=IVCT1,J1
      J2=J-IVCT+1
      IV(J)=IW(J2)
  210 CONTINUE

C     UPDATE IV COUNTER
C

      IVCT=J1
```

```
      C
  215 CONTINUE
C       BYPASS THIS ENTRY IF NR IS ZERO.
C
        IF(NR) 235,235,216
  216 J1=IRVCT+NR
        IF(J1-IRVMX) 227,227,225
  225 WRITE(LUNO,226)
  226 FORMAT(1H ,'TOO MANY VALUES IN RV VECTOR'/)
        CALL EXIT
  227 IRVC1=IRVCT+1
        DO 230 J=IRVC1,J1
        J2= J-IRVCT
        RV(J)=RW(J2)
  230 CONTINUE
C       UPDATE RV COUNTER.
C
        IRVCT=J1
C       STEP INPUT COUNTER.
C
  235 INO=INO+1
        GO TO 35
  300 CONTINUE
C       DECIDE WHETHER IT IS A DISTURBANCE INPUT OR THE LAST
C       INPUT.
C
        IF(NR) 500,500,310
C       IT IS A DISTURBANCE INPUT.STEP DISTURBANCE INPUT
C       COUNTER.
C
  310 IDISC=IDISC+1
C       SET VALUES INTO PROPER VECTORS.
C
        J=IW(2)-100
        NCODE(J)=IW(3)
        VD1(J)=RW(1)
        VD2(J)=RW(2)
        GO TO 35
```

```
      C

C
  400 CONTINUE
C       SPECIAL ENTRIES ARE GIVEN WITH  CODE 0,0 ID NUMBER 1
C       TO 7 AS SHOWN
C       BELOW.FOR EX. A NEW OPTION IS ENTERED AS 0 0 1 8 3 **
C
C
C       1  VECTOR OF OPTIONS.
C       2 CRIT,TO,DELT,DELTH.
C       3  NPTS.
C       4  INITIAL STATE VECTOR X(O).
C       5  VECTOR OF OUTPUT OR STATE VARIABLES NY TO BE DEALT
C       WITH IN THE
C          TIME DOMAIN SOLUTION.
C       6  UTILITY PROGRAM OPTION.
C       7  NUMBER OF JOBS TO BE EXECUTED.
C       8  VACANT.
C
        ISECT=IW(3)
        GO TO(410,463,469,472,475,480,485,490),ISECT
  410 IF(NI-3) 35,35,415
  415 DO 462 K=4,NI,2
        KK=IW(K)
        GO TO(420,425,430,435,440,445,450,455,460,461),KK
  420 NOP(1)=IW(K+1)
        GO TO 462
  425 NOP(2)=IW(K+1)
        GO TO 462
  430 NOP(3)=IW(K+1)
        GO TO 462
  435 NOP(4)=IW(K+1)
        GO TO 462
  440 NOP(5)=IW(K+1)
        GO TO 462
  445 NOP(6)=IW(K+1)
        GO TO 462
  450 NOP(7)=IW(K+1)
        GO TO 462
  455 NOP(8)=IW(K+1)
        GO TO 462
  460 NOP(9)=IW(K+1)
        GO TO 462
  461 NOP(10)=IW(K+1)
  462 CONTINUE
        GO TO 35
```

```
      C
  463 IF(NR) 35,35,464
  464 CRIT=RW(1)
      TO=RW(2)
      DELT=RW(3)
      DELTH=RW(4)
      GO TO 35
  469 IF(NI-3) 35,35,470
  470 NPTS=IW(4)
      GO TO 35
  472 IF(NR) 35,35,473
  473 DO 474 I=1,NR
      X(I)=RW(I)
  474 CONTINUE
      GO TO 35
  475 IF(NI-3) 35,35,476
  476 DO 477 I=4,NI
      NY(I-3)=IW(I)
  477 CONTINUE
      N5=NI-3
      INYFG=INYFG+1
      GO TO 35
  480 IF(NI-3) 35,35,481
  481 NUPOP=IW(4)
      GO TO 35
  485 IF(NI-3) 35,35,486
  486 NJOBS=IW(4)
      GO TO 35
  490 WRITE(LUNO,491)
  491 FORMAT(1H ,'ILLEGAL SPECIAL ENTRY'/)
      GO TO 35
C
  500 INO=INO-1
C     LAST ENTRY GIVES 999 FOR COMPLITION PLUS A NON
C     EXECUTION VALUE
C     OF 0 FOR EXECUTION AND 1 FOR TERMINATION.FOR EX..999,0
C     **,OR 999,1
C
      NEXFG=IW(2)
C     DECIDE WHETHER IT IS A MATRIX INPUT.
```

```
      C
C
      IF(MFLAG) 510,510,600
  510 CALL LINK (LAST2)
C     STEP MATRIX INPUT COUNTER.
C
  600 MREPC=MREPC+1
C     IF IT IS A REPETITION OR CHANGE JUMP TO WAF3.
C
      IF(MREPC-1) 610,610,650
  610 CALL LINK (WAF1)
  650 CALL LINK(WAF2)
      END
```

```
      C
C
C                                   THIS IS LINK 2 OF PART A
C

      EXTERNAL LAST1,LAST3

      DIMENSION IHV(20),NBLKV(30),ITFV(30),ICDV(30),NIV(30)
     1,NRV(30),
     1IV(120),RV(330)
      DIMENSION NY(15),NCODE(10),VD1(10),VD2(10),X(10)
     1,NOP(10)
      DIMENSION IAUX(10)
      DIMENSION IDUM1(80)
      DIMENSION RDUM1(45)
      DIMENSION IDUM2(290),RDUM2(330)

      COMMON IDUM1,RDUM1,IDUM2,RDUM2

      EQUIVALENCE (IDUM1(2),LUNP),(IDUM1(15),NPRTY)
     1,(IDUM1(39),IRVCT)
      EQUIVALENCE   (IDUM1(4),LUNO),(IDUM1(5),N),(IDUM1(6)
     1,M)
      EQUIVALENCE (IDUM1(7),NF),(IDUM1(12),N5),(IDUM1(14)
     1,NPTS)
      EQUIVALENCE (IDUM1(19),NUPOP),(IDUM1(20),INO)
     1,(IDUM1(21),IDISC)
      EQUIVALENCE (IDUM1(18),JOBNO)
      EQUIVALENCE (IDUM1(30),MAXOP),
     1 (IDUM1(32),M1)
      EQUIVALENCE (IDUM1(33),IHMAX),(IDUM1(34),IVMAX)
     1,(IDUM1(35),IRVMX)
      EQUIVALENCE (IDUM1(36),INYFG),
     1 (IDUM1(38),IVCT)
      EQUIVALENCE (IDUM1(41),NJOBS),(IDUM1(42),NEXFG)
      EQUIVALENCE (IDUM1(46),NY(1)),(IDUM1(61),NCODE(1))
     1,(IDUM1(71),NOP
     11))

      EQUIVALENCE (RDUM1(1),CRIT),(RDUM1(2),TO),(RDUM1(3)
     1,DELT)
      EQUIVALENCE (RDUM1(4),DELTH)
      EQUIVALENCE (RDUM1(11),VD1(1))
      EQUIVALENCE (RDUM1(21),VD2(1)),(RDUM1(31),X(1))

      EQUIVALENCE (IDUM2(1),IHV(1)),(IDUM2(21),NBLKV(1))
     1,(IDUM2(51),ITF
     1(1)),(IDUM2(81),ICDV(1)),(IDUM2(111),NIV(1))
     1,(IDUM2(141),NRV(1)),
     2(IDUM2(171),IV(1))
```

```
      C

      EQUIVALENCE (RDUM2(1),RV(1))

      DEFINE FILE 40(95,10,U,NEXT)

C

      WRITE(LUNO,101) JOBNO
  101 FORMAT(1H ,'END OF INPUT DATA FOR JOB NUMBER',I5/)

C     IF INYFG IS STILL ZERO SET NY EQUAL TO Y AND N5 TO M.

      IF(INYFG) 202,202,204
  202 N5=M
      DO 203 I=1,M
      NY(I)=I
  203 CONTINUE

C     IN ADDITION TO M FIND N,NF
C     DETERMINE N.
C

  204 N=0
      DO 210 J=1,IVCT
      IF(IV(J)) 205,210,210
  205 N=N+IABS(IV(J))
  210 CONTINUE

C      DETERMINE NF
C

      NF=0
      DO 230 J=1,IVCT
      ITEM=IABS(IV(J))-100
      IF(ITEM) 230,230,220
  220 NF=NF+1
      IAUX(NF)=ITEM
  230 CONTINUE

C     COMPARE VALUES IN IAUX FOR DUPLICATES.
C

      J1=NF-1
      IF(J1) 300,300,280
  280 DO 290 KK=1,J1
      DO 285 J=KK,J1
      IF(IAUX(KK)-IAUX(J+1)) 285,282,285
  282 NF=NF-1
  285 CONTINUE
  290 CONTINUE
```

```
      C


  300 CONTINUE
C       WRITE THE ENTIRE FILE ON DISK.
C

        K=1
        WRITE(40'K)(IHV(I),I=1,IHMAX),(NBLKV(I),I=1,M1)
       1,(ITFV(I),I=1,M1),
       1ICDV(I),I=1,M1),(NIV(I),I=1,M1),(NRV(I),I=1,M1),(IV(I)
       1,I=1,IVMAX)
       2(RV(I),I=1,IRVMX)

C

        IF(10-NPRTY) 340,340,395
  340 CONTINUE
        READ(40'1) IHV,NBLKV,ITFV,ICDV,NIV,NRV,IV,RV
        WRITE(LUNP,345) M,N,NF
  345 FORMAT(1H ,'M =',I5,5X,'N =',I5,5X,'NF =',I5,5X)
        WRITE(LUNP,351) (IHV(I),I=1,IHMAX)
  351 FORMAT(1H0,'HEAD VECTOR'/(I5))
        WRITE(LUNP,352) (NBLKV(I),ITFV(I),ICDV(I),NIV(I)
       1,NRV(I),I=1,INO)
  352 FORMAT(1H0,'NBLKV ITFV ICDV  NIV  NRV'//(5I5))
        WRITE(LUNP,353) (IV(I),I=1,IVCT)
  353 FORMAT(1H0,'VECTOR IV'/(5I5))
        WRITE(LUNP,354) (RV(I),I=1,IRVCT)
  354 FORMAT(1H0,'VECTOR RV'/(5E15.6))

        IF(IDISC) 395,395,3555

 3555 CONTINUE

        WRITE(LUNP,355) (NOP(I),I=1,MAXOP)
  355 FORMAT(1H0,50X,'NOP'/10I10)
        WRITE(LUNP,356) NPRTY,NPTS,N5,CRIT,TO,DELT,DELTH
  356 FORMAT(1H0,9X,'NPRTY',I5/10X,' NPTS',I5/10X,'   N5',I5
       1/10X,' CRIT
       1,E15.6/10X,'   TO',E15.6/10X,' DELT',E15.6/10X,'DELTH'
       1,E15.6)
        WRITE(LUNP,357) (X(I),I=1,N)
  357 FORMAT(1H0,50X,'X'/10E12.4)
        WRITE(LUNP,358) (NCODE(I),I=1,NF)
  358 FORMAT(1H0,50X,'NCODE'/10I10)
        WRITE(LUNP,359) (VD1(I),I=1,NF)
  359 FORMAT(1H0,50X,'VD1'/10E12.4)
        WRITE(LUNP,360) (VD2(I),I=1,NF)
  360 FORMAT(1H0,50X,'VD2'/10E12.4)
        WRITE(LUNP,361) (NY(I),I=1,N5)
```

```
      C

  361 FORMAT(1H0,50X,'NY'/15I5)
      WRITE(LUNP,362) NUPOP,NJOBS
  362 FORMAT(1H0,9X,'NUPOP',I5/10X,'NJOBS',I5/)

  395 CONTINUE

C
C     CHECK IF FLAG FOR EXECUTION IS ON.
C

      IF(NEXFG) 1450,1450,1400
 1400 WRITE(LUNO,1401)
 1401 FORMAT(1H ,'NO EXECUTION DESIRED'/)

      IF(NUPOP) 1405,1405,1403

 1403 IF(NUPOP-9) 1410,1410,1450

 1405 WRITE(LUNO,1406)
 1406 FORMAT(1H ,'NO UTILITY PROGRAM DESIRED'/)

      NJOBS=NJOBS-1
      IF(NJOBS)  1407,1407,1408
 1407 CALL EXIT
 1408 CALL LINK(LAST1)

 1410 CONTINUE

C     A GO TO STATEMENT WILL BE INSERTED HERE TO
C     BRANCH TO SEVERAL UTILITY PROGRAMS ACCORDING TO NUPOP
C     VALUES FROM 1 TO 9.
C

      WRITE(LUNO,1420)
 1420 FORMAT(1H ,'UTILITY PROGRAM NOT IMPLEMENTED'/)
      CALL EXIT

 1450 CALL LINK(LAST3)
      END
```

```
      C
C
C                                 THIS IS LINK 3 OF PART A
C
      EXTERNAL LAST1,LAST4

      DIMENSION NBLKV(30),ITFV(30),ICDV(30),NIV(30),NRV(30)
     1,IV(120),
     1RV(330)

      DIMENSION IWO(21),RWO(42),IIOV(11),RIOV(10)

      DIMENSION IWV(21),RWV(42),TD(5)

      DIMENSION WCC(60)

      DIMENSION IDUM1(80)
      DIMENSION RDUM1(45)
      DIMENSION IDUM2(290),RDUM2(330)
      DIMENSION IDUM3(70),RDUM3(80)

      COMMON IDUM1,RDUM1,IDUM2,RDUM2,IDUM3,RDUM3

      EQUIVALENCE   (IDUM1(1),LUNR),(IDUM1(2),LUNP)
     1,(IDUM1(3),LUNI)
      EQUIVALENCE (IDUM1(4),LUNO),(IDUM1(13),NTD),(IDUM1(15)
     1,NPRTY)
      EQUIVALENCE (IDUM1(20),INO),(IDUM1(23),IFLG1)
      EQUIVALENCE (IDUM1(24),NFLG1),(IDUM1(25),NFLG2)
     1,(IDUM1(26),NFLG3)

      EQUIVALENCE (RDUM1(6),TD(1))

      EQUIVALENCE (IDUM2(21),NBLKV(1)),(IDUM2(51),ITFV(1))
      EQUIVALENCE (IDUM2(81),ICDV(1)),(IDUM2(111),NIV(1))
      EQUIVALENCE (IDUM2(141),NRV(1)),(IDUM2(171),IV(1))

      EQUIVALENCE (RDUM2(1),RV(1))

      EQUIVALENCE (IDUM3(1),IIOV(1)),(IDUM3(12),IDER)
     1,(IDUM3(13),IPASS)
     1(IDUM3(14),IDDAC),(IDUM3(15),IDNAC),(IDUM3(16),IDD)
     1,(IDUM3(17),
     2IDN),(IDUM3(18),IXBK),(IDUM3(19),NFLGC),(IDUM3(20),IR)
     1,(IDUM3(21)
     3II),(IDUM3(22),NRR),(IDUM3(23),NII),(IDUM3(24),JF)
     1,(IDUM3(25),NBK
     4,(IDUM3(26),IY),(IDUM3(27),IX),(IDUM3(28),IBACK)
      EQUIVALENCE (IDUM3(29),ISPY),(IDUM3(30),IP),(IDUM3(31)
```

```
      C
 1,IN)
  EQUIVALENCE (IDUM3(32),NFLS1),(IDUM3(33),NFLS2)
 1,(IDUM3(34),NFLS3)
  EQUIVALENCE (IDUM3(35),NTDS),(IDUM3(36),ISPTF)
 1,(IDUM3(40),IWV(1))

  EQUIVALENCE (RDUM3(1),RWV(1)),(RDUM3(43),RIOV(1))
  EQUIVALENCE (RDUM3(75),GAIN),(RDUM3(76),DELAY)
 1,(RDUM3(77),TLAG)

  DATA WCC/1.000,1.000,1.208,0.936,0.902,0.985,1.411
 1,0.917,0.490,
 11.084,0.900,1.000,3.333,1.000,0.928,0.946,0.928,0.583
 1,0.984,0.986
 21.644,0.707,1.305,0.959,2.033,0.739,0.859,0.977,1.484
 1,0.680,1.200
 31.000,2.000,1.000,0.500,1.000,1.370,0.950,0.740,0.738
 1,0.365,0.950
 41.435,0.921,1.139,0.749,0.482,1.137,1.495,0.945,0.917
 1,0.771,0.560
 51.006,1.357,0.947,1.176,0.738,0.381,0.995/
C
      IF(IBACK-99) 9,10,9
C     SET INITIAL VALUE COUNTERS.
C
    9 JF=1
      NTD=0
      IX=0
      IY=0
      NFLG1=0
      NFLG2=0
      NFLG3=0
      IDDAC=0
      IDNAC=0
      IXBK=0
      IPASS=0
      IBACK=0
C     SEARCH FOR THE FIRST NON ZERO ENTRY IN VECTOR ITFV.
   10 IBACK=0
      NBK=ITFV(JF)
      IF(NBK) 11,11,19
C     CHECK JF AGAINST INO.
```

```
      C
   11 IF(JF-INO) 12,12,15
   12 JF=JF+1
      GO TO 10
   15 WRITE(LUNO,16)
   16 FORMAT(1H ,'UNSUCCESSFUL INPUT OF DATA')
      CALL EXIT
C     A T.F. IS FOUND AT LOCATION JF.
C     FIND CODE NUMBER
   19 NCD=ICDV(JF)
      CALL FIND (JF,INO,NBLKV,ITFV,NIV,NRV,IV,RV,NII,NRR,IWO
     1,RWO,II,IR,
     1IIOV,RIOV)
      IF(10-NPRTY) 20,20,25
   20 CONTINUE
C
C
C
      WRITE(LUNP,21) (IWO(I),I=1,NII)
      WRITE(LUNP,22) (RWO(I),I=1,NRR)
      WRITE(LUNP,23) (IIOV(I),I=1,II)
      WRITE(LUNP,24) (RIOV(I),I=1,IR)
   21 FORMAT(1H0,'IWO'//(5I5))
   22 FORMAT(1H0,'RWO'//(5E15.6))
   23 FORMAT(1H0,'IIOV'//(5I5))
   24 FORMAT(1H0,'RIOV'//(5E15.6))
C
C
C
   25 CONTINUE
      IF(NCD) 50,50,30
   30 GO TO(60,70,75,80,100,110,120,130),NCD
C     IT IS A STANDARD T.F.
   50 DO 51 K=1,NII
      IWV(K)=IWO(K)
   51 CONTINUE
      DO 52 K=1,NRR
      RWV(K)=RWO(K)
   52 CONTINUE
      GO TO 150
   60 WRITE(LUNO,61)
```

```
      C
 61 FORMAT(1H ,'THIS IS NOT A TRANSFER FUNCTION.NCD IS
   1 ONE')
    CALL EXIT
C        IT IS A STANDARD FIRST ORDER PROCESS T.F.
C        CHECK IF PROCESS TF PARAMETERS ARE GIVEN BY USER OR
C        MUST BE READ
C        FROM DISK FILES.(LIBRARY OF PROCESS T.F. PARAMETERS
C        DETERMINED BY
C        PROGRAMS AS BAKKE'S STEP RESPONSE ANALYSIS ETC.)
C
 70 IF(NRR)72,72,71
 71 IWV(1)=0
    IWV(2)=-1
    RWV(1)=RWO(1)
    RWV(2)=RWO(2)
    RWV(3)=1.0
    RWV(4)=RWO(3)
    NII=2
C        SET THE FLAG FOR PROCESS TRANFER FUNCTION PARAMETERS.
C
    ISPTF=99
    GAIN=RWO(1)
    DELAY=RWO(2)
    TLAG=RWO(3)

    GO TO 150
C        PROCESS T.F. PARAMETERS MAY BE READ AT THIS POINT FROM
C        DISK FILES.
C        PRESENTLY THE FOLLOWING MESSAGE IS GIVEN.
C
 72 WRITE(LUNO,73) NBK
 73 FORMAT(1H ,'PARAMETERS FOR PROCESS T.F.',I5,2X,'NOT
   1 AVAILABLE FRO
   1 LIBRARY'/)
    GO TO 93
C        IT IS A STANDARD SECOND ORDER PROCESS T.F.
C
 75 IF(NRR)72,72,76
 76 IWV(1)=0
    IWV(2)=-1
    IWV(3)=-1
    RWV(1)=RWO(1)
```

```
      C
      RWV(2)=RWO(2)
      RWV(3)=1.0
      RWV(4)=RWO(3)
      RWV(5)=1.0
      RWV(6)=RWO(4)
      NII=3

      GO TO 150

C     IT IS A PROPORTIONAL CONTROLLER T.F.
C

   80 CONTINUE
      IF(NRR) 81,81,98
   81 CONTINUE
      IF(ISPTF-99) 82,84,82
   82 WRITE(LUNO,83) NBK
   83 FORMAT(1H ,'CONTROLLER CONSTANTS FOR T.F.',I5,2X,'NOT
     1 AVAILABLE F
     1OM LIBRARY'/)
      GO TO 93

   84 ISPTF=0

      I2=IWO(1)
      I3=IWO(2)

      DO 85 I=1,3
      RWO(I)=0.0
   85 CONTINUE

      CALL GTCC1(GAIN,DELAY,TLAG,NCD,I2,I3,RWO,WCC)

      IF(3-NPRTY) 86,86,90
   86 CONTINUE
      WRITE(LUNP,87) GAIN,DELAY,TLAG,NCD,I2,I3
   87 FORMAT(1H ,///T16,'CONTROLLER CONSTANTS FROM LIBRARY'/
     1/T2,'PROCES
     1 GAIN..............................',F10.4/T2,'TIME
     1 DELAY..........
     2...................',F10.4/T2,'TIME
     1 LAG.........................
     3.....',F10.4/T2,'CONTROLLER CODE...
     1.......................',I10/T2
     4'CODED CRITERION TO SELECT CONSTANTS.....',I10/T2
     1,'CODED DISTURBA
     5CE TO SELECT CONSTANTS...',I10/)

      WRITE(LUNP,88) (RWO(I),I=1,3)
   88 FORMAT(1H ,'PROPORTIONAL CONSTANT...................'
```

```
      C

     1,F10.4/T2,'I
     1TEGRAL TIME...........................',F10.4/T2
     1,'DERIVATIVE TIME
     2........................',F10.4)

   90 CONTINUE
      IF(NCD-5) 98,105,115
C
C
C     SET FLAG FOR REENTRY TO PREVIOUS LINK.
C
C

   93 IF(LUNR-LUNI) 94,97,94
   94 WRITE(LUNO,95)
   95 FORMAT(1H ,'ENTER CORRECT T.F. PARAMETER'/)

      CALL TWAIT

      IFLG1=99
      INO=INO+1
      CALL LINK (LAST1)

   97 CALL EXIT

   98 IWV(1)=0
      RWV(1)=RWO(1)
      RWV(2)=0.0
      NII=1
      GO TO 150

C     IT IS A PROPORTIONAL PLUS INTEGRAL CONTROLLER T.F.

  100 IF(NRR) 81,81,105
  105 IWV(1)=0
      IWV(2)=1
      IWV(3)=-1
      RWV(1)=RWO(1)
      RWV(2)=0.0
      RWV(3)=1.0
      RWV(4)=RWO(2)
      RWV(5)=0.0
      RWV(6)=RWO(2)
      NII=3
      GO TO 150

C     IT IS A REAL P.I.D. CONTROLLER T.F.
```

```
      C

  110 IF(NRR) 81,81,115
  115 IWV(1)=0
      IWV(2)=2
      IWV(3)=-2
      RWV(1)=RWO(1)
      RWV(2)=0.0
      RWV(3)=1.0
      RWV(4)=RWO(2)
      RWV(5)=RWO(2)*RWO(3)
      RWV(6)=0.2
      RWV(7)=RWO(2)
      RWV(8)=0.2*RWV(5)
      NII=3
      GO TO 150
  120 WRITE(LUNO,121) NBK
  121 FORMAT(1H ,'ILLEGAL INPUT CODE IN T.F.',I5)
      GO TO 93
  130 GO TO 120
C
C     THIS PART DETERMINES THE NUMBER OF FACTORS AT
C     NUMERATOR AND DENOMINATORS PLUS THEIR DEGREES
C

  150 IP=0
      IN=0
      IDN=0
      IDD=0
      NFLGC=0
      NTDS=0
      NFLS1=0
      NFLS2=0
      NFLS3=0
      IDER=0
      K=1
  160 IF(IWV(K)) 161,165,165
  161 IN=IN+1
      IDD=IDD+IABS(IWV(K))
      GO TO 170
  165 IP=IP+1
      IDN=IDN+IWV(K)
  170 IF(K-NII) 171,180,180
  171 K=K+1
      GO TO 160
  180 IDNAC=IDNAC+IDN
      IDDAC=IDDAC+IDD

      IF(10-NPRTY) 181,181,190
  181 CONTINUE
```

```
        C
C
C
      WRITE(LUNP,185) NBK,IP,IN,IDN,IDNAC,IDD,IDDAC
  185 FORMAT(1H0,' NBK    IP    IN   IDN IDNAC IDD IDDAC'//7I5)
C
C
  190 CONTINUE
C     THIS PART CHECKS FOR ILLEGAL TRANSFER FUNCTIONS.
C     1000 PTS ARE ASSUMED FOR DETECTING ANY TIME DELAY.
C
C
      IF(IDN-IDD) 200,230,280
  200 IF(IWV(1)) 299,201,299
  201 IF(ABS(RWV(2))-1.E-03) 299,299,210
  210 NTDS=NTDS+1
      NTD=NTD+NTDS
      TD(NTD)=RWV(2)
      GO TO 299
  230 IF(IN) 240,240,260
  240 NFLGC=NFLGC+1
      IF(ABS(RWV(2))-1.E-03) 2999,2999,250
 2999 ISPY=2

      CALL LINK(LAST5)

  250 NFLS1=NFLS1+1
      NFLG1=NFLG1+NFLS1
      TD(NFLG1)=RWV(2)
      GO TO 299
  260 IF(IWV(1)) 299,270,299
  270 IF(ABS(RWV(2))-1.E-03) 299,299,272
  272 WRITE(LUNO,273) NBK
  273 FORMAT(1H ,'TIME DELAY ASSOCIATED WITH T.F.',I5,5X
     1,'FOR WHICH IDN
     1IDD')
      GO TO 93
  280 IF(IN) 285,285,281
  281 WRITE(LUNO,282) NBK
  282 FORMAT(1H ,'ILLEGAL T.F.',I5,5X,'IDN IS GREATER THAN
     1 IDD')
      GO TO 93
  285 IF(IDN-1) 290,290,286
  286 WRITE(LUNO,287) NBK
  287 FORMAT(1H ,'SECOND ORDER DERIVATIVE TERM IN T.F.',I5
     1,5X,'NOT ALLO
```

```
      C

    1ED')
     GO TO 93
 290 IF(IP-1) 295,295,292
 292 IF(ABS(RWV(2))-1.E-03) 295,295,293
 293 WRITE(LUNO,294) NBK
 294 FORMAT(2H ,'ILLEGAL COMBINATION OF DERIVATIVE TERM AND
    1 T.DELAY IN
    1T.F.',I5)
     GO TO 93
 295 NFLS2=NFLS2+1
     NFLG2=NFLG2+NFLS2
 299 CONTINUE

     IF(10-NPRTY) 3000,3000,3010
3000 CONTINUE

C
C

     WRITE(LUNP,3001)NFLGC,NTDS,NFLS1,NFLS2,NFLS3,NTD,NFLG1
    1,NFLG2,NFLG
3001 FORMAT(1H0,'NFLGC  NTDS NFLS1 NFLS2 NFLS3   NTD NFLG1
    1 NFLG2 NFLG3
    1//9I6)

C
C

3010 CONTINUE

     CALL LINK(LAST4)
     END
```

```
      C
C
C                                 THIS IS LINK 4 OF PART A
C
      EXTERNAL LAST1,LAST5
      DIMENSION VN(11),VD(11),V1(11),V2(11),Z(11),ALPHA(121)
     1,RWV(42)
      DIMENSION IWV(21)
      DIMENSION IDUM1(80)
      DIMENSION RDUM1(45)
      DIMENSION IDUM2(290),RDUM2(330)
      DIMENSION IDUM3(70),RDUM3(80)
      COMMON IDUM1,RDUM1,IDUM2,RDUM2,IDUM3,RDUM3
      EQUIVALENCE   (IDUM1(1),LUNR),(IDUM1(2),LUNP)
     1,(IDUM1(3),LUNI)
      EQUIVALENCE (IDUM1(4),LUNO),(IDUM1(13),NTD),(IDUM1(15)
     1,NPRTY)
      EQUIVALENCE (IDUM1(24),NFLG1),(IDUM1(25),NFLG2)
     1,(IDUM1(26),NFLG3)
      EQUIVALENCE (IDUM1(20),INO),(IDUM1(23),IFLG1)
      EQUIVALENCE (IDUM3(16),IDD),(IDUM3(17),IDN),(IDUM3(25)
     1,NBK)
      EQUIVALENCE (IDUM3(29),ISPY),(IDUM3(30),IP),(IDUM3(31)
     1,IN)
      EQUIVALENCE (IDUM3(32),NFLS1),(IDUM3(33),NFLS2)
     1,(IDUM3(34),NFLS3)
      EQUIVALENCE (IDUM3(12),IDER),(IDUM3(35),NTDS)
     1,(IDUM3(40),IWV(1))
C
      EQUIVALENCE (VN(1),RDUM3(54)),(RDUM3(65),VD(1))
     1,(RDUM3(1),RWV(1))
C
C
C     THIS PART DECOMPOSES MULTITERM TRANSFER FUNCTIONS
C     INTO SINGLE TERM ONES
C     ONES.
C
C
C     CHECK FOR CONSTANT T.F.'S.
C
```

```
      C
      IF(NFLS1) 300,300,499
  300 NFLAG=0
      L=1
      J=1
      ID=IP
C     ZERO VECTORS OF COEFFICIENTS.
C
      K1=IDD+1
      DO 304 K=1,K1
      VN(K)=0.0
      VD(K)=0.0
  304 CONTINUE
  305 IF(ID-1) 306,310,330
  306 IF(NFLAG) 307,307,330
  307 WRITE(LUNO,308) NBK
  308 FORMAT(1H ,'ILLEGAL T.F.',I5,5X,'NUMERATOR IS ZERO')
      GO TO 700
  310 IF(NFLAG) 318,318,315
  315 L1=L+IDD
      DO 316 K=L,L1
      K1=K-L+1
      VD(K1)=RWV(K)
  316 CONTINUE
      GO TO 399
  318 L1=L+IDN
      DO 320 K=L,L1
      VN(K)=RWV(K)
  320 CONTINUE
      IF(IDN) 321,321,322
  321 L=L1+2
      GO TO 370
  322 L=L1+1
      GO TO 370
  330 ID1=IABS(IWV(J))
      L1=L+ID1
      DO 333 K=L,L1
      K1=K-L+1
      V1(K1)=RWV(K)
  333 CONTINUE
      IF(ID1) 335,335,336
  335 L=L1+2
      GO TO 340
  336 L=L1+1
  340 ID=ID-1
  345 J=J+1
      ID2=IABS(IWV(J))
      L1=L+ID2
      DO 348 K=L,L1
```

```
      C
      K1=K-L+1
      V2(K1)=RWV(K)
  348 CONTINUE
      L=L1+1
      IDIM1=ID1+1
      IDIM2=ID2+1
      CALL PMPY (Z,IDIMZ,V1,IDIM1,V2,IDIM2)
      ID=ID-1
      IF(ID) 355,355,350
  350 ID1=IDIMZ-1
      DO 351 K=1,IDIMZ
      V1(K)=Z(K)
  351 CONTINUE
      GO TO 345
  355 IF(NFLAG) 365,365,360
  360 DO 361 K=1,IDIMZ
      VD(K)=Z(K)
  361 CONTINUE
      GO TO 399
  365 DO 366 K=1,IDIMZ
      VN(K)=Z(K)
  366 CONTINUE
  370 J=J+1
      NFLAG=NFLAG+1
      ID=IN
      IF(NFLS2) 305,305,499
  399 CONTINUE

      IF(10-NPRTY) 3998,3998,4000
 3998 CONTINUE

C
C

      IDD1=IDD+1
      WRITE(LUNP,3999) (VN(I),VD(I),I=1,IDD1)
 3999 FORMAT(1H0,5X,'VN',10X,'VD'//(2E15.6))

C
C

 4000 CONTINUE

C     THIS PART BUILDS THE MATRIX ALPHA,FINDS THE
C     COEFFICIENTS B'S AND
C     STORES THEM INTO VN VECTOR,THE COEFFICIENTS A'S BEING
C     STORED IN VD
C
C
```

```
      C
      LD=IDD+1
C       FIND COEFFICIENT OF HIGHEST TERM.
      HT=VD(LD)
C     CHECK IF IT IS ZERO.
      IF(ABS(HT)-1.E-06) 400,400,410
  400 WRITE(LUNO,401) NBK
  401 FORMAT(1H ,'ZERO HIGHEST ORDER COEFFICIENT IN T.F.'
     1,I5)
      CALL EXIT
C       DIVIDE BOTH NUMERATOR AND DENOMINATOR BY HT.
  410 DO 415 K=1,LD
      VN(K)=VN(K)/HT
      VD(K)=VD(K)/HT
  415 CONTINUE
C     ZERO MATRIX ALPHA FROM SECOND COLUMN ON.
      LDLD=LD*LD
      DO 418 K=LD,LDLD
      ALPHA(K)=0.0
  418 CONTINUE
C     SET UP MATRIX ALPHA.
      DO 430 J=1,LD
      J1=LD-J+1
      DO 420 K=1,J1
      K1=LD-K+1
      K2=K+(J-1)*(LD+1)
      ALPHA(K2)=VD(K1)
  420 CONTINUE
  430 CONTINUE
C     REVERSE VECTOR OF BETA'S BEFORE SOLVING THE SYSTEM FOR
C     B'S.
      L1=LD/2
      DO 440 K=1,L1
      TEMP=VN(K)
      K1=LD-K+1
      VN(K)=VN(K1)
      VN(K1)=TEMP
  440 CONTINUE
```

```
      C
      IF(10-NPRTY) 441,441,445
  441 CONTINUE
C
C
      WRITE(LUNP,442)
  442 FORMAT(1H0,10X,'MATRIX ALPHA'/)
      CALL MATPR(LD,LD,ALPHA,Z,LUNP)
C
C
  445 CONTINUE
      CALL SIMQ(ALPHA,VN,LD,IER1)
      IF(10-NPRTY) 460,460,499
  460 CONTINUE
C
C
      WRITE(LUNP,450) (VN(I),I=1,LD)
  450 FORMAT(1H0,'COEFFICIENTS B'/(E15.6))
C
C
C
  499 CONTINUE
C
C     THIS PART ANALYZES DIFFERENT CASES  AND BRANCHES
C     CORRESPONDINGLY.
C
C
      IF(NFLS1) 510,510,540
  510 IF(NFLS2) 511,511,570
  511 IF(NTDS) 512,512,513
  512 ISPY=2
      GO TO 610
  513 GO TO (520,525,535,535),NTD
  520 ISPY=3
C     MATRICES D1 AND E1 NEEDED.
      GO TO 610
  525 ISPY=4
C     MATRICES D2 AND E2 NEEDED.
```

```
      C

    GO TO 610
535 WRITE(LUNO,536)
536 FORMAT(1H ,'ONLY TWO TIME DELAYS ALLOWED'/)
    GO TO 700
540 IF(NTD) 545,545,541
541 WRITE(LUNO,542)
542 FORMAT(1H ,'NO OTHER TIME DELAY ALLOWED WITH AN
   1 ISOLATED ONE'/)
    GO TO 700
545 IF(NFLG1-1) 550,550,546
546 WRITE(LUNO,547)
547 FORMAT(1H ,'ONLY ONE ISOLATED TIME DELAY ALLOWED'/)
    GO TO 700
550 IF(NFLG2) 551,551,552
551 ISPY=1

C       MATRICES AL AND AM ARE NEEDED.

    GO TO 610
552 IF(NFLG2-1) 555,555,553
553 WRITE(LUNO,554)
554 FORMAT(1H ,'ONLY ONE DERIVATIVE ACTION ALLOWED'/)
    GO TO 700
555 NFLS3=NFLS3+1
    NFLG3=NFLG3+NFLS3
    GO TO 551
570 IF(NTD) 575,575,571
571 WRITE(LUNO,572)
572 FORMAT(1H ,'NO TIME DELAY ALLOWED WITH A DERIVATIVE
   1 ACTION'/)
    GO TO 700
575 IF(NFLG2-1) 576,576,553
576 IF(NFLG1) 600,600,582
582 IF(NFLG1-1) 585,585,546
585 NFLS3=NFLS3+1
    NFLG3=NFLG3+NFLS3
600 IDER=IDER+1
610 CONTINUE

    IF(10-NPRTY) 620,620,640

620 CONTINUE
    WRITE(LUNP,630) IDER,ISPY
630 FORMAT(1H ,'IDER ISPY'/2I5)

640 CONTINUE

    CALL LINK(LAST5)

700 CONTINUE
```

```
      C
      IF(LUNR-LUNI) 710,750,710
  710 WRITE(LUNO,715)
  715 FORMAT(1H ,'ENTER CORRECT T.F. PARAMETER'/)
      CALL TWAIT
      IFLG1=99
      INO=INO+1
      CALL LINK(LAST1)
  750 CALL EXIT
      END
```

```
      C
C
C                                  THIS IS LINK 5 OF PART A
C
C      THIS PART SETS VALUES OF A'S AND B'S INTO SPECIFIC
C      LOCATION OF PROPER MATRICES
C
C

       EXTERNAL LAST3,LAST6

       DIMENSION V1(11),VN(11),VD(11),RWV(42),RIOV(10)
      1,IIOV(11)

       DIMENSION A1(100),B1(150),C1(100),F(150),G(225),H(150)
      1,AL(225),
      1AM(150),AL1(225),D1(150),E1(100),D2(150),E2(100)
       DIMENSION A(100),B(200),C(150),D(300)

       DIMENSION IDUM1(80)
       DIMENSION RDUM1(45)
       DIMENSION IDUM2(290),RDUM2(330)
       DIMENSION IDUM3(70),RDUM3(80)
       DIMENSION RDUM4(300)

       COMMON IDUM1,RDUM1,IDUM2,RDUM2,IDUM3,RDUM3,RDUM4

       EQUIVALENCE (IDUM1(2),LUNP),(IDUM1(4),LUNO),(IDUM1(5)
      1,N)
       EQUIVALENCE (IDUM1(6),M),(IDUM1(7),NF),(IDUM1(13),NTD)
       EQUIVALENCE  (IDUM1(15),NPRTY),(IDUM1(20),INO)
      1,(IDUM1(24),NFLG1)
       EQUIVALENCE (IDUM1(29),MMAX)

       EQUIVALENCE(RDUM4(1),A1(1),B1(1),C1(1),F(1),G(1),H(1)
      1,AL(1),AM(1)
      1AL1(1),D1(1),E1(1),D2(1),E2(1))
       EQUIVALENCE (RDUM4(1),A(1),B(1),C(1),D(1))

C
C

       EQUIVALENCE (IDUM3(1),IIOV(1)),(IDUM3(12),IDER)
      1,(IDUM3(13),IPASS)
      1(IDUM3(14),IDDAC),(IDUM3(15),IDNAC),(IDUM3(16),IDD)
      1,(IDUM3(17),
      2IDN),(IDUM3(18),IXBK),(IDUM3(19),NFLGC), (IDUM3(21)
      1,II),
      3(IDUM3(24),JF),(IDUM3(25),NBK)
      4,(IDUM3(26),IY),(IDUM3(27),IX),(IDUM3(28),IBACK)
```

```
      C

      EQUIVALENCE(IDUM3(29),ISPY)

      EQUIVALENCE (RDUM3(1),RWV(1)),(RDUM3(43),RIOV(1))
     1,(RDUM3(54),VN(1
     1),(RDUM3(65),VD(1))

      DEFINE FILE 40(95,10,U,NEXT)
      DEFINE FILE 91(100,2,U,NXT91)
      DEFINE FILE 92(200,2,U,NXT92)
      DEFINE FILE 93(150,2,U,NXT93)
      DEFINE FILE 94(300,2,U,NXT94)
      DEFINE FILE 96(420,2,U,NXT96)
      DEFINE FILE 99(160,10,U,NXT99)

      DEFINE FILE 51(100,2,U,NXT1)
      DEFINE FILE 52(150,2,U,NXT2)
      DEFINE FILE 53(100,2,U,NXT3)
      DEFINE FILE 54(150,2,U,NXT4)
      DEFINE FILE 55(225,2,U,NXT5)
      DEFINE FILE 56(150,2,U,NXT6)
      DEFINE FILE 57(225,2,U,NXT7)
      DEFINE FILE 58(150,2,U,NXT8)
      DEFINE FILE 59(225,2,U,NXT9)
      DEFINE FILE 60(150,2,U,NXT10)
      DEFINE FILE 61(100,2,U,NXT11)
      DEFINE FILE 62(150,2,U,NXT12)
      DEFINE FILE 63(100,2,U,NXT13)

C     ZERO ALL MATRICES THE FIRST TIME THROUGH

      IPASS=IPASS+1
      IF(IPASS-1) 10,10,20
   10 MTOT=300
      DO 15 I=1,MTOT
      RDUM4(I)=0.0
   15 CONTINUE

      WRITE(91'1) A
      WRITE(92'1) B
      WRITE(93'1) C
      WRITE(94'1) D

      WRITE(51'1) A1
      WRITE(52'1) B1
      WRITE(53'1) C1
      WRITE(54'1) F
      WRITE(55'1) G
      WRITE(56'1) H
      WRITE(57'1) AL
      WRITE(58'1) AM
```

```
      C
      WRITE(59'1) AL1
      WRITE(60'1) D1
      WRITE(61'1) E1
      WRITE(62'1) D2
      WRITE(63'1) E2
C
C
   20 IF(IDER) 21,21,800
   21 CONTINUE
      NFLAG=0
      IVR=M
      NID=NBK

      IF(NFLGC) 615,615,625
  615 IF(ABS(VN(1))-1.E-06) 710,710,620
  620 DO 623 K=1,II
      V1(K)=VN(1)*RIOV(K)
  623 CONTINUE
      GO TO 630
  625 DO 628 K=1,II
      V1(K)=RIOV(K)*RWV(1)
  628 CONTINUE
  630 CONTINUE

  635 J=1

C     DISTINGUISH BETWEEN A U AND A Y.

  636 IF(IIOV(J)-100) 675,675,640
  640 KK=IIOV(J)-100
      NZ=NID+IVR*(KK-1)
      GO TO(650,655,660,665),ISPY
  650 WRITE(58'NZ)V1(J)
      GO TO 701
  655 IF(NFLAG) 656,656,658
  656 WRITE(56'NZ)V1(J)
      GO TO 701
  658 WRITE(53'NZ)V1(J)
      GO TO 701
  660 WRITE(61'NZ)V1(J)
      GO TO 701
  665 WRITE(63'NZ)V1(J)
      GO TO 701

  675 KK=IIOV(J)
      NZ=NID+IVR*(KK-1)
      GO TO (680,685,690,695),ISPY
  680 WRITE(57'NZ)V1(J)
```

```
      C

      GO TO 701
  685 IF(NFLAG) 686,686,688
  686 WRITE(55'NZ)V1(J)
      GO TO 701
  688 WRITE(52'NZ)V1(J)
      GO TO 701
  690 WRITE(60'NZ)V1(J)
      GO TO 701
  695 WRITE(62'NZ)V1(J)
  701 IF(J-II) 702,710,710
  702 J=J+1
      GO TO 636
  710 IF(IDD) 760,760,712
  712 IF(NFLAG) 715,715,739
C
C     STEP UP COUNTER FOR X.
C
  715 IX=IX+1
      NZ=NID+M*(IX-1)
      FIND(54'NZ)
      AA=1.0
      WRITE(54'NZ)AA
      IF(IDD-1) 725,725,720
  720 K1=IX+IDD-2
      DO 721 K=IX,K1
      NZ=K+K*N
      WRITE(51'NZ)AA
  721 CONTINUE
      IX=K1+1
  725 DO 730 K=1,IDD
      NZ=IX+N*(IDDAC-IDD)+N*(K-1)
      AB=-VD(K)
      WRITE(51'NZ)AB
  730 CONTINUE
      IVR=N
      NFLAG=NFLAG+1
      JT=2
      NID=IX-IDD
  739 IF(JT-IDD-1) 740,740,760
  740 IF(ABS(VN(JT))-1.E-06) 741,741,742
  741 VN(JT)=0.0
  742 DO 743 K=1,II
      V1(K)=RIOV(K)*VN(JT)
  743 CONTINUE
      JT=JT+1
      NID=NID+1
      GO TO 635
  760 IY=IY+1
      GO TO 1000
```

```
            C
C
  800 J=1
  806 KK=IIOV(J)
      NZ=NBK+M*(KK-1)
      K=1
  807 V1(K)=VN(K)*RIOV(J)
      IF(K-1) 810,810,815
  810 WRITE(55'NZ)V1(K)
      GO TO 820
  815 WRITE(59'NZ)V1(K)
  820 IF(K-IDN-1) 821,825,825
  821 K=K+1
      GO TO 807
  825 IF(J-II) 826,760,760
  826 J=J+1
      GO TO 806
C     FINAL CHECKING.X AND Y COUNTERS ARE COMPARED WITH
C     OVERALL DIMENSIONS N AND M.IF THERE IS AN ERROR
C     EXECUTION IS STOPPED AFTER PRINTING OUT A MESSAGE
C
 1000 CONTINUE
      IF(10-NPRTY) 1400,1400,1450
 1400 CONTINUE
C
C
      WRITE(LUNP,1004) IX,IY
 1004 FORMAT(1H0,'  IX   IY'//2I5)
C
C
 1450 CONTINUE
      IF(IX-N) 1001,1001,1005
 1001 IF(IY-M) 1002,1010,1005
 1002 WRITE(LUNO,1003)
 1003 FORMAT(1H ,'INPUT OF A T.F. AND ITS CONFIGURATION DATA
     1 SET COMPLE
     1ED'/)
      JF=JF+1
      IF(JF-INO) 1040,1040,1007
 1005 WRITE(LUNO,1006)
 1006 FORMAT(1H ,'FINAL ERROR MESSAGE'//'WRONG OVERALL
     1 DIMENSIONS')
      CALL EXIT
 1007 WRITE(LUNO,1008)
```

```
      C
1008 FORMAT(1H ,'UNSUCCESSFUL INPUT OF DATA')
     CALL EXIT
1010 IF(IX-N) 1002,1011,1005
1011 IF(IDNAC-IDDAC) 1020,1025,1026
1020 WRITE(LUNO,1021)
1021 FORMAT(1H ,'DATA TRANSMISSION SUCCESSFUL'/)
     GO TO 1100
1025 IF(NFLG1+NTD) 1020,1020,1026
1026 WRITE(LUNO,1027)
1027 FORMAT(1H ,'FINAL ERROR MESSAGE'//'ILLEGAL OVERALL
    1 CONBINATION OF
    1T.F. AND TIME DELAYS')
     CALL EXIT
1040 IBACK=99
     CALL LINK (LAST3)
1100 CONTINUE
     IF(8-NPRTY) 1150,1150,1350
1150 CONTINUE
C
C
     READ(51'1) A1
     WRITE(LUNP,1200)
1200 FORMAT(1H0,20X,'MATRIX A1'/)
     CALL MATPR(N,N,A1,RWV,LUNP)
     READ(52'1) B1
     WRITE(LUNP,1210)
1210 FORMAT(1H0,20X,'MATRIX B1'/)
     CALL MATPR(N,M,B1,RWV,LUNP)
     READ(53'1) C1
     WRITE(LUNP,1220)
1220 FORMAT(1H0,20X,'MATRIX C1'/)
     CALL MATPR(N,NF,C1,RWV,LUNP)
     READ(54'1) F
     WRITE(LUNP,1230)
1230 FORMAT(1H0,20X,'MATRIX F'/)
     CALL MATPR(M,N,F,RWV,LUNP)
     READ(55'1) G
     WRITE(LUNP,1240)
1240 FORMAT(1H0,20X,'MATRIX G'/)
     CALL MATPR(M,M,G,RWV,LUNP)
     READ(56'1) H
     WRITE(LUNP,1250)
1250 FORMAT(1H0,20X,'MATRIX H'/)
     CALL MATPR(M,NF,H,RWV,LUNP)
     READ(57'1) AL
     WRITE(LUNP,1260)
1260 FORMAT(1H0,20X,'MATRIX AL'/)
     CALL MATPR(M,M,AL,RWV,LUNP)
```

```
C
      READ(58'1) AM
      WRITE(LUNP,1270)
 1270 FORMAT(1H0,20X,'MATRIX AM'/)
      CALL MATPR(M,NF,AM,RWV,LUNP)
      READ(59'1) AL1
      WRITE(LUNP,1280)
 1280 FORMAT(1H0,20X,'MATRIX AL1'/)
      CALL MATPR(M,M,AL1,RWV,LUNP)
      READ(60'1) D1
      WRITE(LUNP,1290)
 1290 FORMAT(1H0,20X,'MATRIX D1'/)
      CALL MATPR(N,M,D1,RWV,LUNP)
      READ(61'1) E1
      WRITE(LUNP,1300)
 1300 FORMAT(1H0,20X,'MATRIX E1'/)
      CALL MATPR(N,NF,E1,RWV,LUNP)
      READ(62'1) D2
      WRITE(LUNP,1310)
 1310 FORMAT(1H0,20X,'MATRIX D2'/)
      CALL MATPR(N,M,D2,RWV,LUNP)
      READ(63'1) E2
      WRITE(LUNP,1320)
 1320 FORMAT(1H0,20X,'MATRIX E2'/)
      CALL MATPR(N,NF,E2,RWV,LUNP)
C
C
C
C
 1350 CONTINUE
      CALL LINK(LAST6)
      END
```

```
      C
C
C                                    THIS IS LINK 6 OF PART A
C
      EXTERNAL LAST7
C     ALL MATRIX MANIPULATIONS ARE CARRIED OUT IN THREE
C     BLOCKS DUM1,DUM2,DUM3 IN COMMON
      DIMENSION A1(100),B1(150),C1(100),F(150),G(225),H(150)
     1,AL(225),
     1AM(150),AL1(225),D1(150),E1(100),D2(150),E2(100)
      DIMENSION UM(225),AT(100),BT(100),CT(150),DT(150)
     1,AL2(225),
     1AM2(150),Q2(225),A2(225),C2(225),B2(150),B3(150)
     1,B4(150),B5(100)
      DIMENSION WA3(100)
      DIMENSION A(100),B(200),C(150),D(300)
      DIMENSION MV(5),MV1(5),NU(20),NU1(20),UD(20),UD1(20)
      DIMENSION W1(15),W2(15)
      DIMENSION IDUM1(80)
      DIMENSION RDUM1(45)
      DIMENSION MBLOK(50),RBLOK(40)
      DIMENSION DUM1(300),DUM2(225),DUM3(225)
      DIMENSION TD(5)
      COMMON IDUM1,RDUM1,MBLOK,RBLOK,DUM1,DUM2,DUM3
      EQUIVALENCE (IDUM1(5),N),(IDUM1(6),M),(IDUM1(7),NF)
      EQUIVALENCE (IDUM1(8),N1),(IDUM1(9),N2),(IDUM1(10),N3)
      EQUIVALENCE (IDUM1(11),N4),(IDUM1(13),NTD)
      EQUIVALENCE (IDUM1(24),NFLG1),(IDUM1(25),NFLG2)
     1,(IDUM1(26),NFLG3)
      EQUIVALENCE (IDUM1(27),IDET),(IDUM1(28),INTD)
      EQUIVALENCE (MBLOK(1),MV(1)),(MBLOK(6),MV1(1))
     1,(MBLOK(11),NU(1)),
     1(MBLOK(31),NU1(1))
      EQUIVALENCE (RBLOK(1),UD(1)),(RBLOK(21),UD1(1))
      EQUIVALENCE (DUM1(1),UM(1),F(1),H(1),AL(1),AM(1)
     1,AL1(1),A2(1),
     1C2(1),B(1),D(1),A(1),C(1),D1(1),D2(1))
```

```
      C
       EQUIVALENCE (DUM2(1),G(1),B1(1),B2(1),B3(1),B4(1)
      1,B5(1))
       EQUIVALENCE (DUM3(1),CT(1),DT(1),AL2(1),AM2(1),Q2(1)
      1,A1(1),AT(1),
      1C1(1),BT(1),E1(1),E2(1),WA3(1))
       EQUIVALENCE (RDUM1(6),TD(1))
       DEFINE FILE 40(95,10,U,NEXT)
       DEFINE FILE 91(100,2,U,NXT91)
       DEFINE FILE 92(200,2,U,NXT92)
       DEFINE FILE 93(150,2,U,NXT93)
       DEFINE FILE 94(300,2,U,NXT94)
       DEFINE FILE 96(420,2,U,NXT96)
       DEFINE FILE 99(160,10,U,NXT99)
       DEFINE FILE 51(100,2,U,NXT1)
       DEFINE FILE 52(150,2,U,NXT2)
       DEFINE FILE 53(100,2,U,NXT3)
       DEFINE FILE 54(150,2,U,NXT4)
       DEFINE FILE 55(225,2,U,NXT5)
       DEFINE FILE 56(150,2,U,NXT6)
       DEFINE FILE 57(225,2,U,NXT7)
       DEFINE FILE 58(150,2,U,NXT8)
       DEFINE FILE 59(225,2,U,NXT9)
       DEFINE FILE 60(150,2,U,NXT10)
       DEFINE FILE 61(100,2,U,NXT11)
       DEFINE FILE 62(150,2,U,NXT12)
       DEFINE FILE 63(100,2,U,NXT13)
C       DEFINE MORE WORKING MATRICES ON DISK FILES.
       DEFINE FILE 70(100,2,U,NXT70)
       DEFINE FILE 71(100,2,U,NXT71)
       DEFINE FILE 72(150,2,U,NXT72)
       DEFINE FILE 73(150,2,U,NXT73)
       DEFINE FILE 74(225,2,U,NXT74)
       DEFINE FILE 75(150,2,U,NXT75)
       DEFINE FILE 76(225,2,U,NXT76)
       DEFINE FILE 77(225,2,U,NXT77)
       DEFINE FILE 78(225,2,U,NXT78)
       DEFINE FILE 79(150,2,U,NXT79)
       DEFINE FILE 80(150,2,U,NXT80)
       DEFINE FILE 81(150,2,U,NXT81)
       DEFINE FILE 82(100,2,U,NXT82)
       DEFINE FILE 83(100,2,U,NXT83)
       DEFINE FILE 84(100,2,U,NXT84)
       DEFINE FILE 85(100,2,U,NXT85)
```

```
      C

      MM=M*M
      MN=M*N
      MNF=M*NF
      NM=MN
      NN=N*N
      NNF=N*NF
C     ZERO FLAGS FOR ZERO DETERMINANT (SINGULAR MATRIX) AND
C     TOO MANY
C     TIME DELAYS.
      IDET=0
      INTD=0
C     SET UP A UNIT MATRIX IN UM OF LENGTH M*M.
      DO 10 I=1,MM
      UM(I)=0.0
   10 CONTINUE
      DO 20 I=1,M
      K=(I-1)*M+I
      UM(K)=1.0
   20 CONTINUE
C     READ G FROM DISK.
      K=1
      READ(55'K)(G(I),I=1,MM)
C     FIND UM-G.
      CALL GMSUB(UM,G,G,M,M)
C     FIND THE INVERSE OF UM-G.
      CALL MINV(G,M,DET,W1,W2)
C     CHECK FOR SINGULAR MATRIX.
      IF(ABS(DET)-1.E-06) 30,30,40
   30 IDET=IDET+1
      GO TO 2000
C     READ MATRIX F FROM DISK.
   40 K=1
      READ(54'K)(F(I),I=1,MN)
C     MULTIPLY THE INVERSE OF (I-G) BY F AND H.
```

```
      C
      CALL GMPRD(G,F,CT,M,M,N)
C     WRITE CT AND DT ON DISK.
      K=1
      WRITE(72'K)(CT(I),I=1,MN)
      K=1
      READ(56'K)(H(I),I=1,MNF)
      CALL GMPRD(G,H,DT,M,M,NF)
      K=1
      WRITE(73'K)(DT(I),I=1,MNF)
      IF(NFLG1) 45,45,41
   45 IF(NFLG2) 49,49,47
   41 READ(57'1) (AL(I),I=1,MM)
      CALL GMPRD(G,AL,AL2,M,M,M)
      WRITE(74'1) (AL2(I),I=1,MM)
      READ(58'1) (AM(I),I=1,MNF)
      CALL GMPRD(G,AM,AM2,M,M,NF)
      WRITE(75'1) (AM2(I),I=1,MNF)
      IF(NFLG3) 49,49,47
   47 READ(59'1) (AL1(I),I=1,MM)
      CALL GMPRD(G,AL1,Q2,M,M,M)
      WRITE(76'1) (Q2(I),I=1,MM)
   49 CONTINUE
C     READ B1 FROM DISK.
      K=1
      READ(52'K)(B1(I),I=1,NM)
C     READ CT FROM DISK.
      K=1
      READ(72'K)(CT(I),I=1,MN)
C     MULTIPLY B1 BY CT AND BY DT.
      CALL GMPRD(B1,CT,A2,N,M,N)
C     READ A1 FROM DISK AND ADD IT TO A2.
      K=1
      READ(51'K)(A1(I),I=1,NN)
      CALL GMADD(A1,A2,AT,N,N)
      K=1
      WRITE(70'K)(AT(I),I=1,NN)
      K=1
      READ(73'K)(DT(I),I=1,MNF)
      CALL GMPRD(B1,DT,C2,N,M,NF)
```

```
      C

C     READ C1 FROM DISK AND ADD IT TO C2.

      K=1
      READ(53'K)(C1(I),I=1,NNF)
      CALL GMADD(C1,C2,BT,N,NF)
      K=1
      WRITE(71'K)(BT(I),I=1,NNF)

      IF(NFLG2) 60,60,1000
   60 IF(NFLG1) 100,100,112

  100 IF(NTD) 110,110,130

C     COMPRESS MATRICES DT AND BT BEFORE STORING THEM AS D
C     AND B.

  110 TDV=0.0
      N1=0
      N2=0
      CALL MATCO(TDV,N,NF,BT,N2,B,N1,MV,NU,UD)
      K=1
      NN2=N*N2
      WRITE(92'K)(B(I),I=1,NN2)
  111 TDV=0.0
      N3=0
      N4=0
      K=1
      READ(73'K)(DT(I),I=1,MNF)
      CALL MATCO(TDV,M,NF,DT,N4,D,N3,MV1,NU1,UD1)
      K=1
      NN4=N*N4
      WRITE(94'1)(D(I),I=1,NN4)

  112 CONTINUE

C     TRANFER AT INTO A AND CT INTO C.

      K=1
      READ(70'K)(AT(I),I=1,NN)
      DO 115 I=1,NN
      A(I)=AT(I)
  115 CONTINUE
      WRITE(91'1)(A(I),I=1,NN)
      K=1
      READ(72'K)(CT(I),I=1,MN)
      DO 120 I=1,MN
      C(I)=CT(I)
  120 CONTINUE
      WRITE(93'1)(C(I),I=1,MN)
```

```
      C
      GO TO 1000
  130 K=1
      READ(60'K)(D1(I),I=1,NM)
C     MULTIPLY D1 BY CT AND DT.
      K=1
      READ(72'K)(CT(I),I=1,MN)
      CALL GMPRD(D1,CT,B2,N,M,N)
      K=1
      WRITE(79'K)(B2(I),I=1,NN)
      K=1
      READ(73'K)(DT(I),I=1,MNF)
      CALL GMPRD(D1,DT,B3,N,M,NF)
C     READ E1 FROM DISK AND ADD IT TO B3.
      K=1
      READ(61'K)(E1(I),I=1,NNF)
      CALL GMADD(B3,E1,B3,N,NF)
      K=1
      WRITE(80'K)(B3(I),I=1,NNF)
      IF(NTD-1) 140,140,150
C     COMPACT BT,B3,B2 INTO B.
  140 TDV=0.0
      N1=0
      N2=0
      K=1
      READ(71'K)(BT(I),I=1,NNF)
      CALL MATCO(TDV,N,NF,BT,N2,B,N1,MV,NU,UD)
      TDV=TD(1)
      CALL MATCO(TDV,N,NF,B3,N2,B,N1,MV,NU,UD)
      K=1
      READ(79'K)(B2(I),I=1,NN)
      CALL MATCO(TDV,N,N,B2,N2,B,N1,MV,NU,UD)
      K=1
      NN2=N*N2
      WRITE(92'K)(B(I),I=1,NN2)
      GO TO 111
C     READ D2 FROM DISK AND MULTIPLY IT BY CT AND DT.
  150 K=1
      READ(62'K)(D2(I),I=1,NM)
      K=1
      READ(72'K)(CT(I),I=1,MN)
```

```
      C
      CALL GMPRD(D2,CT,B4,N,M,N)
      K=1
      WRITE(81'K)(B4(I),I=1,NN)
      READ(73'1)(DT(I),I=1,MNF)
      CALL GMPRD(D2,DT,B5,N,M,NF)
C     READ E2 FROM DISK AND ADD IT TO B5.
      K=1
      READ(63'K)(E2(I),I=1,NNF)
      CALL GMADD(E2,B5,B5,N,NF)
      K=1
      WRITE(82'K)(B5(I),I=1,NNF)
      IF(NTD-2) 160,160,170
C     COMPACT BT,B3,B2,B5,B4 INTO B MATRIX.
  160 TDV=0.0
      N1=0
      N2=0
      K=1
      READ(71'K)(BT(I),I=1,NNF)
      CALL MATCO(TDV,N,NF,BT,N2,B,N1,MV,NU,UD)
      TDV=TD(1)
      K=1
      READ(80'K)(B3(I),I=1,NNF)
      CALL MATCO(TDV,N,NF,B3,N2,B,N1,MV,NU,UD)
      K=1
      READ(79'K)(B2(I),I=1,NN)
      CALL MATCO(TDV,N,N,B2,N2,B,N1,MV,NU,UD)
      TDV=TD(2)
      K=1
      READ(82'K)(B5(I),I=1,NNF)
      CALL MATCO(TDV,N,NF,B5,N2,B,N1,MV,NU,UD)
      K=1
      READ(81'K)(B4(I),I=1,NN)
      CALL MATCO(TDV,N,N,B4,N2,B,N1,MV,NU,UD)
      K=1
      NN2=N*N2
      WRITE(92'K)(B(I),I=1,NN2)
      GO TO 111
  170 INTD=INTD+1
      GO TO 2000
 1000 CONTINUE
      IF(NFLG2) 2000,2000,1100
C     THIS PART PROVIDES AN INVERTED MATRIX USED IN THE NEXT
```

```
      C
C     LINK.
 1100 CONTINUE
C     TRANFER Q2 INTO A2.
      READ(76'1)(Q2(I),I=1,MM)
      DO 210 I=1,MM
      A2(I)=Q2(I)
  210 CONTINUE
C     MULTIPLY Q2 BY CT AND CALL THE PRODUCT B3.
      READ(72'1)(CT(I),I=1,MN)
      CALL GMPRD(A2,CT,B3,M,M,N)
      WRITE(80'1)(B3(I),I=1,MN)
      READ(52'1)(B1(I),I=1,NM)
      DO 290 I=1,NM
      A2(I)=B1(I)
  290 CONTINUE
      READ(80'1)(B3(I),I=1,MN)
      CALL GMPRD(A2,B3,WA3,N,M,N)
      WRITE(85'1)(WA3(I),I=1,NN)
C     SET UP A UNIT MATRIX IN B5.
      DO 350 I=1,NN
      B5(I)=0.0
  350 CONTINUE
      DO 360 I=1,N
      K=(I-1)*N+I
      B5(K)=1.0
  360 CONTINUE
      READ(85'1)(WA3(I),I=1,NN)
      CALL GMSUB(B5,WA3,B5,N,N)
      CALL MINV(B5,N,DET,W1,W2)
      IF(ABS(DET)-1.E-06) 30,30,1500
 1500 WRITE(82'1)(B5(I),I=1,NN)
 2000 CALL LINK(LAST7)
      END
```

```
      C
C
C                                THIS IS LINK 7 OF PART A
C
      EXTERNAL WAF2,WAF3

      DIMENSION A1(100),B1(150),C1(100),F(150),G(225),H(150)
     1,AL(225),
     1AM(150),AL1(225),D1(150),E1(100),D2(150),E2(100)
      DIMENSION UM(225),AT(100),BT(100),CT(150),DT(150)
     1,AL2(225),
     1AM2(150),Q2(225),A2(225),C2(225),B2(150),B3(150)
     1,B4(150),B5(100),
     2WA1(100),WA2(100)
      DIMENSION A(100),B(200),C(150),D(300)
      DIMENSION MV(5),MV1(5),NU(20),NU1(20),UD(20),UD1(20)
      DIMENSION WW(30),W1(15),W2(15)
      DIMENSION NOP(10)
      DIMENSION IDUM1(80)
      DIMENSION RDUM1(45)
      DIMENSION MBLOK(50),RBLOK(40)
      DIMENSION DUM1(300),DUM2(225),DUM3(225)
      DIMENSION TD(5)

      COMMON IDUM1,RDUM1,MBLOK,RBLOK,DUM1,DUM2,DUM3

      EQUIVALENCE (IDUM1(2),LUNP),(IDUM1(4),LUNO)
      EQUIVALENCE (IDUM1(5),N),(IDUM1(6),M),(IDUM1(7),NF)
      EQUIVALENCE (IDUM1(8),N1),(IDUM1(9),N2),(IDUM1(10),N3)
      EQUIVALENCE (IDUM1(11),N4),(IDUM1(13),NTD),(IDUM1(15)
     1,NPRTY)
      EQUIVALENCE (IDUM1(24),NFLG1),(IDUM1(25),NFLG2)
     1,(IDUM1(26),NFLG3)
      EQUIVALENCE (IDUM1(27),IDET),(IDUM1(28),INTD)
     1,(IDUM1(21),IDISC)
      EQUIVALENCE (IDUM1(71),NOP(1))
      EQUIVALENCE (MBLOK(1),MV(1)),(MBLOK(6),MV1(1))
     1,(MBLOK(11),NU(1)),
     1(MBLOK(31),NU1(1))
      EQUIVALENCE (RBLOK(1),UD(1)),(RBLOK(21),UD1(1))
      EQUIVALENCE (DUM1(1),UM(1),F(1),H(1),AL(1),AM(1)
     1,AL1(1),A2(1),
     1C2(1),B(1),D(1),A(1),C(1),D1(1),D2(1))
      EQUIVALENCE (DUM2(1),G(1),B1(1),B2(1),B3(1),B4(1)
     1,B5(1))
      EQUIVALENCE (DUM3(1),CT(1),DT(1),AL2(1),AM2(1),Q2(1)
     1,A1(1),AT(1),
     1C1(1),BT(1),E1(1),E2(1),WA1(1),WA2(1))
      EQUIVALENCE (WW(1),W1(1)),(WW(16),W2(1))
```

```
C
      EQUIVALENCE (RDUM1(6),TD(1))
      DEFINE FILE 40(95,10,U,NEXT)
      DEFINE FILE 91(100,2,U,NXT91)
      DEFINE FILE 92(200,2,U,NXT92)
      DEFINE FILE 93(150,2,U,NXT93)
      DEFINE FILE 94(300,2,U,NXT94)
      DEFINE FILE 96(420,2,U,NXT96)
      DEFINE FILE 99(160,10,U,NXT99)
      DEFINE FILE 51(100,2,U,NXT1)
      DEFINE FILE 52(150,2,U,NXT2)
      DEFINE FILE 53(100,2,U,NXT3)
      DEFINE FILE 54(150,2,U,NXT4)
      DEFINE FILE 55(225,2,U,NXT5)
      DEFINE FILE 56(150,2,U,NXT6)
      DEFINE FILE 57(225,2,U,NXT7)
      DEFINE FILE 58(150,2,U,NXT8)
      DEFINE FILE 59(225,2,U,NXT9)
      DEFINE FILE 60(150,2,U,NXT10)
      DEFINE FILE 61(100,2,U,NXT11)
      DEFINE FILE 62(150,2,U,NXT12)
      DEFINE FILE 63(100,2,U,NXT13)
C       DEFINE MORE WORKING MATRICES ON DISK FILES.
      DEFINE FILE 70(100,2,U,NXT70)
      DEFINE FILE 71(100,2,U,NXT71)
      DEFINE FILE 72(150,2,U,NXT72)
      DEFINE FILE 73(150,2,U,NXT73)
      DEFINE FILE 74(225,2,U,NXT74)
      DEFINE FILE 75(150,2,U,NXT75)
      DEFINE FILE 76(225,2,U,NXT76)
      DEFINE FILE 77(225,2,U,NXT77)
      DEFINE FILE 78(225,2,U,NXT78)
      DEFINE FILE 79(150,2,U,NXT79)
      DEFINE FILE 80(150,2,U,NXT80)
      DEFINE FILE 81(150,2,U,NXT81)
      DEFINE FILE 82(100,2,U,NXT82)
      DEFINE FILE 83(100,2,U,NXT83)
      DEFINE FILE 84(100,2,U,NXT84)
      DEFINE FILE 85(100,2,U,NXT85)
      MM=M*M
      MN=M*N
      MNF=M*NF
      NM=MN
      NN=N*N
      NNF=N*NF
      IF(IDET) 90,90,80
```

```
      C

   80 WRITE(LUNO,81)
   81 FORMAT(1H ,'MATRIX (I-G) IS SINGULAR -- PROGRAM STOPS'
     1/)
      CALL EXIT
   90 IF(INTD) 95,95,92
   92 WRITE(LUNO,93)
   93 FORMAT(1H ,'TOO MANY TIME DELAYS')
      CALL EXIT

   95 IF(NFLG3) 96,96,200
   96 IF(NFLG1) 100,100,250
  100 IF(NFLG2) 1000,1000,200

  200 CONTINUE

C     TRANFER Q2 INTO A2.

      READ(76'1)(Q2(I),I=1,MM)
      DO 210 I=1,MM
      A2(I)=Q2(I)
  210 CONTINUE

C     MULTIPLY Q2 BY DT AND CALL THE PRODUCT B2.

      READ(73'1)(DT(I),I=1,MNF)
      CALL GMPRD(A2,DT,B2,M,M,NF)

C     MATRIX B2 MUST BE A ZERO MATRIX.

      DO 220 I=1,MNF
      IF(ABS(B2(I))-1.E-06) 220,220,215
  215 WRITE(LUNO,216)
  216 FORMAT(1H ,'ILLEGAL POSITION OF DERIVATIVE ACTION')
      CALL EXIT
  220 CONTINUE

  250 IF(NFLG3) 251,251,255
  251 IF(NFLG2) 255,255,300
  255 CONTINUE

      READ(74'1) (AL2(I),I=1,MM)
      DO 260 I=1,MM
      A2(I)=AL2(I)
  260 CONTINUE
      READ(72'1)(CT(I),I=1,MN)
      CALL GMPRD(A2,CT,B2,M,M,N)
      WRITE(79'1)(B2(I),I=1,MN)
      READ(73'1)(DT(I),I=1,MNF)
      CALL GMPRD(A2,DT,B4,M,M,NF)
```

```
      C
      READ(75'1) (AM2(I),I=1,MNF)
      CALL GMADD(AM2,B4,B4,M,NF)
      WRITE(81'1)(B4(I),I=1,MNF)
      IF(NFLG3) 270,270,280
C     COMPACT DT,B4,B2 INTO D.
  270 TDV=0.0
      N3=0
      N4=0
      READ(73'1)(DT(I),I=1,MNF)
      CALL MATCO(TDV,M,NF,DT,N4,D,N3,MV1,NU1,UD1)
      TDV=TD(1)
      READ(81'1)(B4(I),I=1,MNF)
      CALL MATCO(TDV,M,NF,B4,N4,D,N3,MV1,NU1,UD1)
      READ(79'1)(B2(I),I=1,MN)
      CALL MATCO(TDV,M,N,B2,N4,D,N3,MV1,NU1,UD1)
      MN4=M*N4
      WRITE(94'1)(D(I),I=1,MN4)
  280 CONTINUE
      READ(52'1)(B1(I),I=1,NM)
      DO 290 I=1,NM
      A2(I)=B1(I)
  290 CONTINUE
      READ(79'1)(B2(I),I=1,MN)
      CALL GMPRD(A2,B2,WA1,N,M,N)
      WRITE(83'1)(WA1(I),I=1,NN)
      READ(81'1)(B4(I),I=1,MNF)
      CALL GMPRD(A2,B4,WA2,N,M,NF)
      WRITE(84'1)(WA2(I),I=1,NNF)
  300 CONTINUE
      IF(NFLG3) 310,310,320
  310 IF(NFLG2) 311,311,320
  311 TDV=0.0
      N1=0
      N2=0
      READ(71'1)(BT(I),I=1,NNF)
      CALL MATCO( TDV,N,NF,BT,N2,B,N1,MV,NU,UD)
      TDV=TD(1)
      READ(84'1)(WA2(I),I=1,NNF)
      CALL MATCO(TDV,N,NF,WA2,N2,B,N1,MV,NU,UD)
      READ(83'1)(WA1(I),I=1,NN)
      CALL MATCO(TDV,N,N,WA1,N2,B,N1,MV,NU,UD)
      NN2=N*N2
      WRITE(92'1)(B(I),I=1,NN2)
      GO TO 1000
```

```
      C

  320 CONTINUE
      READ(82'1)(B5(I),I=1,NN)
      READ(70'1)(AT(I),I=1,NN)
      CALL GMPRD(B5,AT,A,N,N,N)
      WRITE(91'1)(A(I),I=1,NN)
      READ(71'1)(BT(I),I=1,NNF)
      CALL GMPRD(B5,BT,A2,N,N,NF)
      WRITE(77'1)(A2(I),I=1,NNF)

      IF(NFLG3) 400,400,420
  400 DO 410 I=1,NNF
      E1(I)=A2(I)
  410 CONTINUE
      TDV=0.0
      N1=0
      N2=0
      CALL MATCO(TDV,N,NF,E1,N2,B,N1,MV,NU,UD)
      NN2=N*N2
      WRITE(92'1)(B(I),I=1,NN2)
      GO TO 450
  420 CONTINUE
      READ(83'1)(WA1(I),I=1,NN)
      CALL GMPRD(B5,WA1,C2,N,N,N)
      WRITE(78'1)(C2(I),I=1,NN)
      READ(84'1)(WA2(I),I=1,NNF)
      CALL GMPRD(B5,WA2,D2,N,N,NF)
      WRITE(62'1)(D2(I),I=1,NNF)

C     COMPACT MATRICES A2,D2,C2.

      TDV=0.0
      N1=0
      N2=0
      READ(77'1)(A2(I),I=1,NNF)
      CALL MATCO(TDV,N,NF,A2,N2,G,N1,MV,NU,UD)
      TDV=TD(1)
      READ(62'1)(D2(I),I=1,NNF)
      CALL MATCO(TDV,N,NF,D2,N2,G,N1,MV,NU,UD)
      READ(78'1)(C2(I),I=1,NN)
      CALL MATCO(TDV,N,N,C2,N2,G,N1,MV,NU,UD)

C     TRANFER G INTO B.

      NN2=N*N2
      DO 430 I=1,NN2
      B(I)=G(I)
  430 CONTINUE
      WRITE(92'1)(B(I),I=1,NN2)

  450 CONTINUE
```

```
      C

      READ(80'1)(B3(I),I=1,MN)
      READ(91'1)(A(I),I=1,NN)
      CALL GMPRD(B3,A,AL2,M,N,N)
C     ADD CT TO AL2.
C     FIRST TRANSFER AL2 TO G.
      DO 460 I=1,MN
      G(I)=AL2(I)
  460 CONTINUE
      READ(72'1)(CT(I),I=1,MN)
      CALL GMADD(G,CT,C,M,N)
      WRITE(93'1)(C(I),I=1,MN)
      READ(80'1)(B3(I),I=1,MN)
      READ(77'1)(A2(I),I=1,NNF)
      CALL GMPRD(B3,A2,AM2,M,N,NF)
C     ADD DT TO AM2.
C     FIRST  TRANSFER AM2 TI G.
      DO 470 I=1,MNF
      G(I)=AM2(I)
  470 CONTINUE
      READ(73'1)(DT(I),I=1,MNF)
      CALL GMADD(G,DT,G,M,NF)
      IF(NFLG3) 500,500,510
C     COMPACT G INTO D.
  500 TDV=0.0
      N3=0
      N4=0
      CALL MATCO(TDV,M,NF,G,N4,D,N3,MV1,NU1,UD1)
      MN4=M*N4
      WRITE(94'1)(D(I),I=1,MN4)
      GO TO 1000
  510 WRITE(55'1)(G(I),I=1,MNF)
      READ(80'1)(B3(I),I=1,MN)
      READ(78'1)(C2(I),I=1,NN)
      CALL GMPRD(B3,C2,AL2,M,N,N)
C     ADD B2 TO AL2.
      READ(79'1)(B2(I),I=1,MN)
      CALL GMADD(B2,AL2,B2,M,N)
      WRITE(79'1)(B2(I),I=1,MN)
```

```
C
      READ(80'1)(B3(I),I=1,MN)
      READ(62'1)(D2(I),I=1,NNF)
      CALL GMPRD(B3,D2,AL2,M,N,NF)
C     ADD B4 TO AL2.
      READ(81'1)(B4(I),I=1,MNF)
      CALL GMADD(B4,AL2,B4,M,NF)
      WRITE(81'1)(B4(I),I=1,MNF)
C     COMPACT G,B4,B2 INTO D.
      TDV=0.0
      N3=0
      N4=0
      READ(55'1)(G(I),I=1,MNF)
      CALL MATCO(TDV,M,NF,G,N4,D,N3,MV1,NU1,UD1)
      TDV=TD(1)
      READ(81'1)(B4(I),I=1,MNF)
      CALL MATCO(TDV,M,NF,B4,N4,D,N3,MV1,NU1,UD1)
      READ(79'1)(B2(I),I=1,MN)
      CALL MATCO(TDV,M,N,B2,N4,D,N3,MV1,NU1,UD1)
      MN4=M*N4
      WRITE(94'1)(D(I),I=1,MN4)
 1000 CONTINUE
C     DECIDE IF THE USER WANTS TO OUTPUT THE FINAL MATRICES
C     AND VECTORS.
      IF(10-NPRTY) 1010,1010,1500
 1010 CONTINUE
      READ(91'1) A
      WRITE(LUNP,1015)
 1015 FORMAT(1H0,50X,'MATRIX A'/)
      CALL MATPR(N,N,A,WW,LUNP)
      READ(92'1) B
      WRITE(LUNP,1020)
 1020 FORMAT(1H0,50X,'MATRIX B'/)
      CALL MATPR(N,N2,B,WW,LUNP)
      READ(93'1) C
      WRITE(LUNP,1025)
 1025 FORMAT(1H0,50X,'MATRIX C'/)
      CALL MATPR(M,N,C,WW,LUNP)
      READ(94'1) D
      WRITE(LUNP,1030)
 1030 FORMAT(1H0,50X,'MATRIX D'/)
      CALL MATPR(M,N4,D,WW,LUNP)
```

```
      C
      WRITE(LUNP,1035) (MV(I),I=1,N1)
 1035 FORMAT(1H0,50X,'MV'/5I20)
      WRITE(LUNP,1040) (NU(I),I=1,N2)
 1040 FORMAT(1H0,50X,'NU'/20I5)
      WRITE(LUNP,1045) (UD(I),I=1,N2)
 1045 FORMAT(1H0,50X,'UD'/10E12.4)
      WRITE(LUNP,1050) (MV1(I),I=1,N3)
 1050 FORMAT(1H0,50X,'MV1'/5I20)
      WRITE(LUNP,1055) (NU1(I),I=1,N4)
 1055 FORMAT(1H0,50X,'NU1'/20I5)
      WRITE(LUNP,1060) (UD1(I),I=1,N4)
 1060 FORMAT(1H0,50X,'UD1'/10E12.4)

 1500 CONTINUE

C     CHECK IF VECTOR MV1(I) IS ZERO IN ORDER TO RESET
C     OPTION NO.7.

      ISEE=0
      DO 1510 I=1,N3
      ISEE=ISEE+MV1(I)
 1510 CONTINUE
      IF(ISEE) 1520,1520,1540
 1520 NOP(7)=1
 1540 CONTINUE

C     CHECK IF SPECIAL ENTRIES ARE GIVEN HENCE CONTINUATION
C     IS DESIRED.

      IF(IDISC) 1550,1550,1600
 1550 WRITE(LUNO,1551)
 1551 FORMAT(1H ,'PROGRAM STOPS AS NO DISTURBANCE HAS BEEN
     1 GIVEN'/)
      CALL EXIT

C     CONTINUATION IS DESIRED.

 1600 CONTINUE

C     IF THERE IS A PURE TIME DELAY SET NTD TO 1

      IF(NFLG1) 1650,1650,1640
 1640 NTD=1
 1650 CONTINUE
      CALL LINK(WAF2)
      END
```

```
      C
C
C                              THIS IS LINK 1 OF PART B
C
      EXTERNAL WAF2,WAF3

      DIMENSION IW(25),RW(50),IALPH(1)
      DIMENSION A(100),B(200),C(150),D(300)
      DIMENSION MV(5),MV1(5),NU(20),NU1(20),UD(20),UD1(20)
      DIMENSION TD(5)
      DIMENSION NY(15),NOP(10)
      DIMENSION IDUM1(80)
      DIMENSION RDUM1(45)
      DIMENSION MBLOK(50),RBLOK(40)
      DIMENSION RTEMP(300)

      COMMON IDUM1,RDUM1,MBLOK,RBLOK,RTEMP

      EQUIVALENCE (IDUM1(3),LUNI),(IDUM1(4),LUNO),(IDUM1(5)
     1,N)
      EQUIVALENCE (IDUM1(6),M),(IDUM1(7),NF),(IDUM1(8),N1)
      EQUIVALENCE (IDUM1(9),N2),(IDUM1(10),N3),(IDUM1(11)
     1,N4)
      EQUIVALENCE (IDUM1(12),N5),(IDUM1(13),NTD),(IDUM1(15)
     1,NPRTY)
      EQUIVALENCE (IDUM1(36),INYFG)
      EQUIVALENCE (IDUM1(18),JOBNO),(IDUM1(2),LUNP)
      EQUIVALENCE (IDUM1(46),NY(1)),(IDUM1(71),NOP(1))
      EQUIVALENCE (RDUM1(6),TD(1))
      EQUIVALENCE (MBLOK(1),MV(1)),(MBLOK(6),MV1(1))
     1,(MBLOK(11),NU(1)),
     1(MBLOK(31),NU1(1))
      EQUIVALENCE (RBLOK(1),UD(1)),(RBLOK(21),UD1(1))
      EQUIVALENCE (RTEMP(1),A(1),B(1),C(1),D(1))

      DEFINE FILE 40(95,10,U,NEXT)
      DEFINE FILE 91(100,2,U,NXT91)
      DEFINE FILE 92(200,2,U,NXT92)
      DEFINE FILE 93(150,2,U,NXT93)
      DEFINE FILE 94(300,2,U,NXT94)

C     REQUEST TO ENTER DATA IF INPUT VIA TYPEWRITER.

      ISEE=0
      IF(LUNO-LUNP) 20,10,20
   10 ISEE=1
   20 CONTINUE

      IF(ISEE) 504,504,506
```

```
      C

  504 CONTINUE
      WRITE(LUNO,505)
  505 FORMAT(1H ,'ENTER N M NF'/)
  506 CONTINUE

      CALL TWAIT

      NR=-1
      CALL FFIOR(IW,RW,IALPH,NI,NR,NA,LUNO,LUNI)
      N=IW(1)
      M=IW(2)
      NF=IW(3)

      IF(ISEE) 508,508,511
  508 CONTINUE
      WRITE(LUNO,510)
  510 FORMAT(1H ,'ENTER NUMBER OF TIME DELAYS'/)
  511 CONTINUE

      CALL TWAIT

      NR=-1
      CALL FFIOR(IW,RW,IALPH,NI,NR,NA,LUNO,LUNI)
      NTD=IW(1)
C
C     IF NTD=0 THEN VECTORS MV,NU,UD,MV1,NU1,UD1 ARE SET BY
C     THE PROGRAM.
C     IN SUCH A CASE THE FORCING FUNCTION VECTORS U AND V
C     ARE IDENTICAL.
C
      IF(NTD) 520,520,600
  520 N1=1
      N2=NF
      MV(N1)=N2

      DO 530 I=1,N2
      NU(I)=I
      UD(I)=0.0
  530 CONTINUE

      IF(NOP(7)-1) 700,700,540
  540 N3=1
      N4=NF
      MV1(N3)=N4
      DO 550 I=1,N4
      NU1(I)=I
      UD1(I)=0.0
  550 CONTINUE
      GO TO 700
```

```
      C
600 CONTINUE
    IF(ISEE) 609,609,611
609 CONTINUE
    WRITE(LUNO,610)
610 FORMAT(1H ,'ENTER VECTOR MV(N1)'/)
611 CONTINUE

    CALL TWAIT

    NR=-1
    CALL FFIOR(MV,RW,IALPH,N1,NR,NA,LUNO,LUNI)

    IF(ISEE) 619,619,621
619 CONTINUE
    WRITE(LUNO,620)
620 FORMAT(1H ,'ENTER VECTORS NU(N2) AND UD(N2)'/)
621 CONTINUE

    CALL TWAIT

    NR=-1
    CALL FFIOR(NU,UD,IALPH,N2,NR,NA,LUNO,LUNI)

    IF(NOP(7)-1) 700,700,640

640 IF(ISEE) 641,641,651
641 CONTINUE
    WRITE(LUNO,650)
650 FORMAT(1H ,'ENTER VECTOR MV1(N3)'/)
651 CONTINUE

    CALL TWAIT

    NR=-1
    CALL FFIOR(MV1,RW,IALPH,N3,NR,NA,LUNO,LUNI)

    IF(ISEE) 659,659,661
659 CONTINUE
    WRITE(LUNO,660)
660 FORMAT(1H ,'ENTER VECTORS NU1(N4) AND UD1(N4)'/)
661 CONTINUE

    CALL TWAIT

    NR=-1
    CALL FFIOR(NU1,UD1,IALPH,N4,NR,NA,LUNO,LUNI)

700 MTYPE=0
```

```
      C
      IF(ISEE) 702,702,710
  702 CONTINUE
      WRITE(LUNO,705)
  705 FORMAT(1H ,'ENTER MATRIX A COLUMNWISE'/)
  710 ICT=1
      MTYPE=MTYPE+1
      CALL TWAIT
  720 NR=-1
      CALL FFIOR(IW,RW,IALPH,NI,NR,NA,LUNO,LUNI)
      K1=ICT+NR-1
      DO 770 K=ICT,K1
      KK=K-ICT+1
      GO TO (730,740,750,760),MTYPE
  730 A(K)=RW(KK)
      GO TO 770
  740 B(K)=RW(KK)
      GO TO 770
  750 C(K)=RW(KK)
      GO TO 770
  760 D(K)=RW(KK)
  770 CONTINUE
      ICT=ICT+NR
      GO TO (810,820,830,840),MTYPE
  810 IF(ICT-N*N) 720,815,815
  815 WRITE(91'1) A
      IF(ISEE) 816,816,710
  816 CONTINUE
      WRITE(LUNO,818)
  818 FORMAT(1H ,'ENTER MATRIX B COLUMNWISE'/)
      GO TO 710
  820 IF(ICT-N*N2) 720,825,825
  825 WRITE(92'1) B
      IF(NOP(7)-1) 900,826,826
  826 IF(ISEE) 827,827,710
  827 WRITE(LUNO,828)
  828 FORMAT(1H ,'ENTER MATRIX C COLUMNWISE'/)
      GO TO 710
  830 IF(ICT-M*N) 720,835,835
```

```
      C

  835 WRITE(93'1) C
      IF(NOP(7)-1) 900,900,836

  836 IF(ISEE) 837,837,710
  837 WRITE(LUNO,838)
  838 FORMAT(1H ,'ENTER MATRIX D COLUMNWISE'/)
      GO TO 710

  840 IF(ICT-M*N4) 720,870,870
  870 WRITE(94'1) D

  900 WRITE(LUNO,910) JOBNO
  910 FORMAT(1H ,/1X,'END OF INPUT DATA FOR JOB NUMBER',I5/)

      IF(INYFG) 920,920,950
  920 N5=M
      DO 925 I=1,N5
      NY(I)=I
  925 CONTINUE

  950 CONTINUE
      CALL LINK(WAF2)
      END
```

```
      C
C
C                                   THIS IS LINK 2 OF PART B
C
      EXTERNAL WAF3

      DIMENSION NOP(10),NCODE(10),NY(15),VD1(10),VD2(10)
     1,X(10),TD(5)
      DIMENSION MV(5),MV1(5),NU(20),NU1(20),UD(20),UD1(20)
      DIMENSION A(100),B(200),C(150),D(300)
      DIMENSION RV(20)
      DIMENSION IDUM1(80)
      DIMENSION RDUM1(45)
      DIMENSION MBLOK(50),RBLOK(40)
      DIMENSION RTEMP(300)

      COMMON IDUM1,RDUM1,MBLOK,RBLOK,RTEMP

      EQUIVALENCE   (IDUM1(1),LUNR),(IDUM1(2),LUNP)
     1,(IDUM1(3),LUNI)
      EQUIVALENCE   (IDUM1(4),LUNO),(IDUM1(5),N),(IDUM1(6)
     1,M)
      EQUIVALENCE   (IDUM1(7),NF),(IDUM1(8),N1),(IDUM1(9)
     1,N2)
      EQUIVALENCE   (IDUM1(10),N3),(IDUM1(11),N4),(IDUM1(12)
     1,N5)
      EQUIVALENCE   (IDUM1(13),NTD),(IDUM1(14),NPTS)
     1,(IDUM1(15),NPRTY)
      EQUIVALENCE(IDUM1(18),JOBNO),(IDUM1(30),MAXOP)
      EQUIVALENCE (IDUM1(19),NUPOP),(IDUM1(41),NJOBS)
      EQUIVALENCE (IDUM1(46),NY(1)),(IDUM1(61),NCODE(1))
     1,(IDUM1(71),NOP
     11))
      EQUIVALENCE (RDUM1(1),CRIT),(RDUM1(2),TO),(RDUM1(3)
     1,DELT)
      EQUIVALENCE (RDUM1(4),DELTH)
      EQUIVALENCE (RDUM1(6),TD(1))
      EQUIVALENCE (RDUM1(11),VD1(1))
      EQUIVALENCE (RDUM1(21),VD2(1)),(RDUM1(31),X(1))
      EQUIVALENCE (RBLOK(1),UD(1)),(RBLOK(21),UD1(1))
      EQUIVALENCE (MBLOK(1),MV(1)),(MBLOK(6),MV1(1))
     1,(MBLOK(11),NU(1)),
     1(MBLOK(31),NU1(1))
      EQUIVALENCE (RTEMP(1),A(1),B(1),C(1),D(1))

      DEFINE FILE 40(95,10,U,NEXT)
      DEFINE FILE 91(100,2,U,NXT91)
      DEFINE FILE 92(200,2,U,NXT92)
      DEFINE FILE 93(150,2,U,NXT93)
```

```
      C
      DEFINE FILE 94(300,2,U,NXT94)
C     BYPASS THIS LINK IF NPRTY IS LESS THAN 3
      IF(3-NPRTY)5,5,500
    5 CONTINUE
      K=NOP(1)
      IF(K) 10,10,20
   10 WRITE(LUNP,11)
   11 FORMAT(1H ,///20X,'BLOCK DIAGRAM INPUT'//)
      GO TO 30
   20 WRITE(LUNP,21)
   21 FORMAT(1H ,///24X,'MATRIX INPUT'//)
   30 CONTINUE
      WRITE(LUNP,40) K,JOBNO,NPRTY,LUNI,LUNO
   40 FORMAT(T2,'INPUT MATRIX FLAG.......................'
     1,I10/T2,'JOB
     1UMBER.................................',I10/T2,'PRIORITY
     1 NUMBER FOR
     2RINTING OUTPUT.....',I10/T2,'LOGICAL UNIT NUMBER FOR
     1 INPUT DEVICE
     3...',I10/T2,'LOGICAL UNIT NUMBER FOR OUTPUT DEVICE...'
     1,I10/)
      DO 50 I=1,MAXOP
      WRITE(LUNP,55) I,NOP(I)
   55 FORMAT(T2,'OPTION..',I2,'..........
     1....................',I10)
   50 CONTINUE
      WRITE(LUNP,77)
      VCRIT=10.0**(-CRIT)*100.0
      WRITE(LUNP,60) VCRIT,TO,DELT,DELTH,NPTS
   60 FORMAT(T2,'ABSOLUTE CRITERION VALUE (PER CENT).....'
     1,F10.4/T2,'IN
     1TIAL TIME VALUE.......................',F10.4/T2,'TIME
     1 INTERVAL...
     2........................',F10.4/T2,'TIME
     1 SUBINTERVAL..............
     3.........',F10.4/T2,'NUMBER OF TIME
     1 INTERVALS.................',I1
     4/)
      DO 75 I=1,N
      WRITE(LUNP,70) I,X(I)
   70 FORMAT(T2,'INITIAL STATE VARIABLE..',I2
     1,'..............',F10.4)
```

```
      C
  75 CONTINUE
     WRITE(LUNP,77)
     DO 90 I=1,N5
     WRITE(LUNP,85) I,NY(I)
  85 FORMAT(T2,'DESIRED OUTPUT/STATE VARIABLE..',I2
    1,'.......',I10)
  90 CONTINUE
     WRITE(LUNP,77)
     WRITE(LUNP,100) NUPOP,NJOBS
 100 FORMAT(T2,'UTILITY PROGRAM NUMBER..................'
    1,I10/T2,'NUMB
    1R OF JOBS...........................',I10/)
     DO 110 I=1,NF
     WRITE(LUNP,105) I,NCODE(I),VD1(I),VD2(I)
 105 FORMAT(T2,'DIST..',I2,'  CODE..',I2,'  1ST PAR..'
    1,F10.4,'  2ND PA
    1..',F10.4)
 110 CONTINUE
     WRITE(LUNP,120)
 120 FORMAT(1H ,///T11,'COEFFICIENT MATRICES OF THE STATE
    1 EQUATION'/T2
    1'AND RELATED DATA'//)
     WRITE(LUNP,130) N,M,NF,NTD
 130 FORMAT(T2,'NUMBER OF STATE VARIABLES...............'
    1,I10/T2,'NUMB
    1R OF OUTPUT VARIABLES..............',I10/T2,'NUMBER OF
    1 EXTERNAL F
    2RCING FUNCTIONS....',I10/T2,'NUMBER OF TIME
    1 DELAYS...............
    3...',I10/)
     WRITE(LUNP,140) MV(1)
 140 FORMAT(T2,'NUMBER OF EXTERNAL FORCING FUNCTIONS IN
    1 U(T)     ',I2)
     IF(N1-3) 180,150,150
 150 DO 160 I=2,N1,2
     I1=I+1
     WRITE(LUNP,170) MV(I),MV(I1)
 170 FORMAT(T2,'DELAYED EXTERNAL FORCING FUNCTIONS
    1 IN...U(T)     ',I2/T
    1,'DELAYED STATE VARIABLES IN..............U(T)     '
    1,I2)
```

```
      C

160 CONTINUE
180 CONTINUE

    WRITE(LUNP,77)
 77 FORMAT(1H )

    DO 200 I=1,N2
    WRITE(LUNP,190) NU(I),UD(I)
190 FORMAT(T2,'VARIABLE IN U(T)....',I2,'...DELAY
   1 VALUE....',F10.4)
200 CONTINUE
    WRITE(LUNP,77)

    IF(NOP(7)-1) 280,280,210
210 WRITE(LUNP,215) MV1(1)
215 FORMAT(T2,'NUMBER OF EXTERNAL FORCING FUNCTIONS IN
   1 V(T)     ',I2)

    IF(N3-3) 250,220,220
220 DO 230 I=2,N3,2
    I1=I+1
    WRITE(LUNP,225) MV1(I),MV1(I1)
225 FORMAT(T2,'DELAYED EXTERNAL FORCING FUNCTIONS
   1 IN...V(T)     ',I2/T
   1,'DELAYED STATE VARIABLES IN.............V(T)     '
   1,I2)
230 CONTINUE
250 CONTINUE
    WRITE(LUNP,77)

    DO 270 I=1,N4
    WRITE(LUNP,260) NU1(I),UD1(I)
260 FORMAT(T2,'VARIABLE IN V(T)....',I2,'...DELAY
   1 VALUE....',F10.4)
270 CONTINUE
280 CONTINUE

    READ(91'1)A
    WRITE(LUNP,300)
300 FORMAT(1H ,/T27,'MATRIX A'/)
    CALL MATPR(N,N,A,RV,LUNP)

    READ(92'1)B
    WRITE(LUNP,320)
320 FORMAT(1H ,/T27,'MATRIX B'/)
    CALL MATPR(N,N2,B,RV,LUNP)

    IF(NOP(7)-1) 500,350,350
350 WRITE(LUNP,360)
360 FORMAT(1H ,/T27,'MATRIX C'/)
```

```
      C
      READ(93'1)C
      CALL MATPR(M,N,C,RV,LUNP)

      IF(NOP(7)-1) 500,500,400
  400 WRITE(LUNP,410)
  410 FORMAT(1H ,/T27,'MATRIX D'/)
      READ(94'1)D
      CALL MATPR(M,N4,D,RV,LUNP)

  500 CONTINUE
C     READ MATRIX A INTO COMMON BEFORE MOVING TO NEXT LINK.

      READ(91'1)A
      CALL LINK(WAF3)
      END
```

```
     C
C
C                                 THIS IS LINK 3 OF PART B
C

      EXTERNAL WAF4

      DIMENSION A(100)
      DIMENSION EVR(10),EVI(10)
      DIMENSION WORK1(100),WORK2(10),WORK3(11),WORK4(11)
     1,WORK5(11),
     1WORK6(11)
      DIMENSION IDUM1(80),RDUM1(45),MBLOK(50),RBLOK(40)
     1,RTEMP(300),
     1RW(40)
      DIMENSION NOP(10)

      COMMON IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW(10),RW

      EQUIVALENCE (A(1),RTEMP(1)),(RW(1),EVR(1)),(RW(11)
     1,EVI(1))
      EQUIVALENCE (IDUM1(2),LUNP),(IDUM1(5),N),(IDUM1(15)
     1,NPRTY)
      EQUIVALENCE (IDUM1(4),LUNO)
      EQUIVALENCE (IDUM1(44),MPSFG),(IDUM1(45),MEVCT)
     1,(IDUM1(71),NOP(1)
      EQUIVALENCE (RTEMP(101),WORK1(1))

      MEVCT=0
      MPSFG=0

C     DECIDE ABOUT OPTION 2.

      IF(NOP(2)) 100,100,50
   50 CONTINUE

C     THE RUNGA-KUTTA METHOD IS CALLED IN TO INTEGRATE
C     DIRECTTLY THE SYSTEM OF DIFFERENTIAL EQUATIONS.
C     AT THIS POINT A CALL WILL BE INSERTED TO A SPECIAL
C     PROGRAM THAT FINDS THE TIME DOMAIN SOLUTION OF THE
C     MATRIX DIFFERENTIAL EQUATION.THE PROGRAM WILL END
C     WITH A CALL LINK(WAF8).THE C-H TECHNIQUE IS
C     ENTIRELY BYPASSED.PRESENTLY THE FOLLOWING STATE-
C     MENT IS PRINTED OUT.

      WRITE(LUNP,51)
   51 FORMAT(1H ,'RUNGA-KUTTA METHOD FOR DIRECT INTEGRATION
     1 NOT IMPLEME
     1TED'/)
      CALL EXIT
```

```
      C
  100 CONTINUE
C     DECIDE ABOUT OPTION 3.
      IF(NOP(3)) 120,120,110
  110 CONTINUE
C     A FLAG IS SET AT THIS POINT FOR PROPER ENTRY TO LINK
C     WAF5.
      MPSFG=99
      CALL LINK (WAF5)
  120 CONTINUE
C     DECIDE WHICH METHOD MUST BE USED TO FIND EIGENVALUES.
C     EACH METHOD WILL CONSIST OF A NUMBER OF SUBROUTINES
C     MENTIONED IN LOCAL CONTROL CARDS.THE INPUT COMMON TO
C     ALL METHODS IS THE COEFFICIENT MATRIX A,THE COMMON
C     OUTPUT TWO VECTORS (EVR,EVI) CONTAINING REAL AND IMAG-
C     INARY PARTS OF EIGENVALUES.IF THE METHOD PROVIDES THE
C     MATRIX OF THE EIGENVECTORS THIS WILL BE MATRIX RMOD
C     (100) DIMENSIONED AND EQUIVALENCED TO RTEMP AS
C     (RTEMP(201),RMOD(1)).IF THIS IS NOT THE CASE NOP(5)
C     MUST BE SET TO ZERO.WORK1 TO WORK6 VECTORS WILL BE
C     AVAILABLE AS WORKING VECTORS.
      KOP5=NOP(5)
      K=NOP(4)
      GO TO(121,121,230,280),K
  121 CONTINUE
C     HERE STARTS METHOD USING SUBROUTINES CHEQN+BART.
      CALL CHEQN(N,A,EVR,EVI,WORK1,WORK2,WORK3,IER1)
      IF(IER1) 125,125,122
  122 WRITE(LUNO,123) IER1
  123 FORMAT(1H ,/1X,'ERROR CODE IN SUBROUTINE CHEQN IS',I5
     1/1X,'TRY POW
     1R SERIES METHOD'/)
      CALL EXIT
  125 CONTINUE
      IF(10-NPRTY) 131,131,133
  131 WRITE(LUNP,132) IER1
  132 FORMAT(1H ,10X,'IER1'//10X,I5/)
```

```
      C
  133 CONTINUE
      IF(5-NPRTY) 134,134,140
  134 K=N+1
      WRITE(LUNP,77)
   77 FORMAT(1H ,/)
      DO 138 I=1,K
      J=I-1
      WRITE(LUNP,135) J,WORK3(I)
  135 FORMAT(T2,'COEFFICIENT OF POWER...',I2,'  OF CHAR.
     1 EQUATION......
     1,E12.4)
  138 CONTINUE
  140 CONTINUE
      DO 145 I=1,N
      EVR(I)=0.0
      EVI(I)=0.0
  145 CONTINUE
      IF(NOP(4)-1) 141,141,180
  141 CONTINUE
C     GIVE NUMBER OF ITERATION FOR SUB. BART
      LIMI=250
      CALL BART(N,WORK3,WORK5,WORK6,EVR,EVI,WORK4,LIMI,IER2)
      IF(IER2) 148,148,146
  146 WRITE(LUNO,147) IER2
  147 FORMAT(1H ,/1X,'ERROR CODE IN SUBROUTINE BART IS',I5
     1/1X,'TRY POWE
     1 SERIES METHOD'/)
      CALL EXIT
  148 CONTINUE
      IF(10-NPRTY) 170,170,175
  170 WRITE(LUNP,171) (WORK4(I),I=1,K)
  171 FORMAT(1H0,10X,'LAST VALUES OF APROXIMATING
     1 POLYNOMIAL'//(10X,
     1E15.6)/)
  175 CONTINUE
  149 CONTINUE
C     RESET OPTION NUMBER 5 AS THIS METHOD DOES NOT PROVIDE
C     EIGENVECTORS
      NOP(5)=0
```

```
      C

      IF(10-NPRTY) 150,150,152
  150 WRITE(LUNP,151) IER2
  151 FORMAT(1H ,10X,'IER2'//10X,I5/)
  152 CONTINUE

      IF(5-NPRTY) 160,160,165
  160 CONTINUE

      WRITE(LUNP,78)
   78 FORMAT(1H )

      DO 164 I=1,N
      WRITE(LUNP,161) EVR(I),EVI(I)
  161 FORMAT(T2,'REAL PART OF EI.V.',E12.4,'    IMAG. PART OF
     1 EI.V.',E12
     14)
  164 CONTINUE

  165 CONTINUE

      GO TO 300

  180 CONTINUE

C     THIS OPTION USES SUBROUTINES PRBM AND PQFB

      IC=N+1

      CALL PRBM (WORK3,IC,EVR,EVI,WORK4,IR,IER2)

      IF(IER2) 188,188,182
  182 WRITE(LUNO,183) IER2
  183 FORMAT(1H ,/1X,'ERROR CODE IN SUBROUTINE PRBM IS',I5
     1/1X,'TRY POWE
     1 SERIES METHOD'/)
      CALL EXIT
  188 CONTINUE

      IF(10-NPRTY) 190,190,195
  190 ICT=IC-IR
      WRITE(LUNP,191) IR,(WORK4(I),I=1,ICT)
  191 FORMAT(1H0,10X,'NUMBER OF CALCULATED ROOTS',I5/9X
     1,'COEFFICIENTS O
     1REMAINDER POLYNOMIAL'/(10X,E15.6)/)
  195 CONTINUE

      GO TO 149

  230 WRITE(LUNP,231)
  231 FORMAT(1H ,'METHOD TO FIND EIGENVALUES NOT AVAILABLE'
```

```
      C

    1/)
     CALL EXIT
 280 GO TO 230
 300 CONTINUE
C
C     DECIDE ABOUT OPTION 5.
C
     IF(KOP5) 310,310,302
 302 IF(NOP(5)) 303,303,305
 303 WRITE(LUNO,304)
 304 FORMAT(1H ,'DESIRED METHOD USING EIGENVECTORS NOT
    1 APPLICABLE'/1X,
    1'PROGRAM IS CONTINUED IN THE STANDARD WAY'/)
     GO TO 310
 305 CONTINUE
C
C     SET A FLAG AND PROCEED.
C
     MEVCT=99
     WRITE(LUNO,308)
 308 FORMAT(1H ,/1X,'METHOD USING EIGENVECTORS NOT
    1 IMPLEMENTED'/1X,'TR
    1 ANOTHER ONE'/)
     CALL EXIT

 310 CONTINUE
     CALL LINK(WAF4)
     END
```

```
          C
C
C                                   THIS IS LINK 4 OF PART B
C
      EXTERNAL WAF5

      DIMENSION EVR(10),EVI(10),EVR1(10),EVI1(10),CC(100)
     1,MT(5)
      DIMENSION IDUM1(80),RDUM1(45),MBLOK(50),RBLOK(40)
     1,RTEMP(300),
     1IW(10),RW(40)
      DIMENSION WR(10),WI(10),WORK1(10),WORK2(10)
      DIMENSION NOP(10)

      COMMON IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW

      EQUIVALENCE (IDUM1(2),LUNP),(IDUM1(5),N),(IDUM1(15)
     1,NPRTY)
      EQUIVALENCE (IDUM1(4),LUNO),(IDUM1(71),NOP(1))
      EQUIVALENCE (RDUM1(1),CRIT),(RDUM1(3),DELT),(RDUM1(5)
     1,SMAST)
      EQUIVALENCE (RDUM1(4),DELTH)
      EQUIVALENCE (IW(1),L),(IW(2),ND),(IW(3),NR),(IW(4),NC)
      EQUIVALENCE (IW(6),MT(1))
      EQUIVALENCE (RW(1),EVR(1)),(RW(11),EVI(1))
      EQUIVALENCE (RW(21),EVR1(1)),(RW(31),EVI1(1))
      EQUIVALENCE (RTEMP(101),CC(1))

      K=NOP(6)

C     TRANSFER EVR AND EVI INTO WORKING VECTORS WORK1 AND
C     WORK2

      DO 3 I=1,N
      WORK1(I)=EVR(I)
      WORK2(I)=EVI(I)
    3 CONTINUE

      CALL EIGCK(K,N,WORK1,WORK2,WR,WI,NPR,EVR1,EVI1,CRIT
     1,DELT,DELTH,
     1SMAST,LUNO,L,IER3)

      IF(IER3) 310,310,305
  305 WRITE(LUNO,306) IER3
  306 FORMAT(1H ,/1X,'ERROR CODE IN SUBROUTINE EIGCK IS',I5
     1/1X,'TRY POW
     1R SERIES METHOD'/)
      CALL EXIT
  310 CONTINUE
```

```
      C
      IF(10-NPRTY) 320,320,330
  320 WRITE(LUNP,321) IER3
  321 FORMAT(1H1,10X,'IER3'//10X,I5/)
  330 CONTINUE
      IF(5-NPRTY) 340,340,350
  340 CONTINUE
      WRITE(LUNP,10)
      DO 343 I=1,L
      WRITE(LUNP,341) WORK1(I),WORK2(I)
  341 FORMAT(T2,'REAL PART OF SHIFTED EI.V.',E12.4,'   IMAG.
     1 PART.',E12
     14)
  343 CONTINUE
      WRITE(LUNP,10)
   10 FORMAT(1H )
      WRITE(LUNP,344) SMAST
  344 FORMAT(T2,'REAL PART OF SMALLEST EI
     1.V.........................',E1
     1.4/)
      IF(NPR) 350,350,345
  345 CONTINUE
      DO 348 I=1,NPR
      WRITE(LUNP,347) EVR1(I),EVI1(I)
  347 FORMAT(T2,'REAL PART OF POSITIVE EI.V.',E12.4,'  IMAG.
     1 PART.',E12
     14)
  348 CONTINUE
      WRITE(LUNP,10)
  350 CONTINUE
      IF(5-NPRTY) 360,360,370
  360 CONTINUE
      DO 363 I=1,L
      WRITE(LUNP,361) EVR1(I),EVI1(I)
  361 FORMAT(T2,'REAL PART OF ORDERED EI.V.',E12.4,'   IMAG.
     1 PART.',E12
     14)
  363 CONTINUE
```

```
      C
      WRITE(LUNP,10)
      WRITE(LUNP,364) ND,NR,NC
  364 FORMAT(T2,'NUMBER OF DISTINCT EI.VS
     1...........................',I1
     1/T2,'NUMBER OF REPEATED EI.VS......
     1....................',I10/T2,'
     2UMBER OF COMPLEX EI.VS...........................',I10
     1/)
      IF(NR)370,370,365
  365 CONTINUE
      DO 368 I=1,NR
      WRITE(LUNP,367) MT(I)
  367 FORMAT(T2,'MULTEPLICITY OF REPEATED
     1 EI.VS....................',I1
     1)
  368 CONTINUE
      WRITE(LUNP,10)
  370 CONTINUE
C     RECALCULATE ORIGINAL REAL PART OF EI.VS TO BE USED
C     IN BUILDING MATRIX CC.
      DO 380 I=1,L
      WR(I)=EVR1(I)+ SMAST
      WI(I)=EVI1(I)
  380 CONTINUE
      CALL BLDMC(L,ND,NR,MT,NC,WR,WI,WORK1,WORK2,CC,IER4)
      IF(10-NPRTY) 400,400,410
  400 WRITE(LUNP,401)     IER4
  401 FORMAT(1H0,10X,'IER4'//10X,I5/)
  410 CONTINUE
      IF(5-NPRTY) 420,420,430
  420 WRITE(LUNP,421)
  421 FORMAT(1H ,/T27,'MATRIX CC'/)
      CALL MATPR(L,L,CC,WORK1,LUNP)
  430 CONTINUE
      CALL LINK (WAF5)
      END
```

```
          C
C
C                                   THIS IS LINK 5 OF PART B
C
      EXTERNAL WAF6

      DIMENSION A(100),CC(100),FM(100),FMR(100),EVR(10)
     1,EVI(10),EVR1(10
      DIMENSION H(200),EVI1(10)
      DIMENSION WORK1(100),WORK7(100)
      DIMENSION IDUM1(80),RDUM1(45),MBLOK(50),RBLOK(40)
     1,RTEMP(300)
      DIMENSION IW(10),RW(40)
      DIMENSION MT(5)

      COMMON IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW,FM,H

      EQUIVALENCE (IDUM1(2),LUNP),(IDUM1(5),N),(IDUM1(15)
     1,NPRTY)
      EQUIVALENCE (IDUM1(44),MPSFG)
      EQUIVALENCE (IDUM1(4),LUNO)
      EQUIVALENCE (RDUM1(1),CRIT),(RDUM1(3),DELT),(RDUM1(4)
     1,DELTH)
      EQUIVALENCE (RDUM1(5),SMAST)
      EQUIVALENCE (IW(1),L),(IW(2),ND),(IW(3),NR),(IW(4),NC)
     1,
     1(IW(6),MT(1))
      EQUIVALENCE (RW(1),EVR(1)),(RW(11),EVI(1))
      EQUIVALENCE (RW(21),EVR1(1)),(RW(31),EVI1(1))
      EQUIVALENCE (RTEMP(1),A(1)),(RTEMP(101),CC(1))
      EQUIVALENCE (H(1),FMR(1))

      K1=0
      NPS=DELT/DELTH+0.5
      NN=N*N

C     CHECK FOR MPSFG AND MEVCT.

  400 CONTINUE
      T=DELTH*FLOAT(K1)
      IF(MPSFG-99) 460,565,460
  460 CONTINUE

      CALL BLDVV(L,EVR1,EVI1,ND,NR,MT,NC,CRIT,DELTH,EVI,K1)

      IF(5-NPRTY) 500,500,510
  500 CONTINUE

      WRITE(LUNP,10)
```

```
      C

   10 FORMAT(1H ,/)

      DO 508 I=1,L
      WRITE(LUNP,501) T,I,EVI(I)
  501 FORMAT(T2,'TIME.....',F10.4,'  EXPONENTIAL OF EI.V('
     1,I2,').......'
     1E12.4)
  508 CONTINUE

      WRITE(LUNP,11)
   11 FORMAT(1H )

  510 CONTINUE

C     TRANSFER MATRIX CC INTO WORKING VECTOR WORK1.

      LL=L*L
      DO 515 K=1,LL
      WORK1(K)=CC(K)
  515 CONTINUE

      CALL SIMQ(WORK1,EVI,L,IER1)

      IF(10-NPRTY) 520,520,530
  520 WRITE(LUNP,521) IER1
  521 FORMAT(1H0,10X,'IER1'/10X,I5/)
  530 CONTINUE

      IF(5-NPRTY) 540,540,550
  540 CONTINUE

      DO 548 I=1,L
      J=I-1
      WRITE(LUNP,541) T,J,EVI(I)
  541 FORMAT(T2,'TIME.....',F10.4,'  COEFFICIENT ALPHA(',I2
     1,').........'
     1E12.4)
  548 CONTINUE

      WRITE(LUNP,11)

  550 CONTINUE

      S=EXP(SMAST*T)

C     INITIATE WORK7 TO MATRIX A AND FM TO ZERO.

      DO 480 K=1,NN
      WORK7(K)=A(K)
      FM(K)=0.0
```

```
      C
  480 CONTINUE
C         CALCULATE THE FUNDAMENTAL MATRIX AT DELTH.
      DO 560 K=1,L
      TEMP=EVI(K)
      CALL BLDMR(N,A,WORK7,WORK1,K)
      CALL MSMUL(N,N,WORK1,TEMP,WORK1)
      CALL GMADD(WORK1,FM,FM,N,N)
  560 CONTINUE
C     CORRECT FM BY S FACTOR
      CALL MSMUL(N,N,FM,S,FM)
      GO TO 569
  565 CONTINUE
C     THIS IS THE POWER SERIES METHOD
C     GIVE THE NUMBER OF ITERATIONS
C
      NIT=100
      CALL POSRT(N,A,FM,WORK1,WORK7,NIT,DELTH,K1,IER)
C     IF THE SERIES DOES NOT CONVERGE IN NIT ITERATIONS
C     THE PROGRAM STOPS
      IF(IER) 569,569,566
  566 WRITE(LUNO,567) NIT
  567 FORMAT(1H ,/1X,'POWER SERIES METHOD DOES NOT CONVERGE
     1 IN',I5,'ITE
     1ATIONS'/)
      CALL EXIT
  569 IF(5-NPRTY) 570,570,580
  570 WRITE(LUNP,571) T
  571 FORMAT(1H ,/T14,'FUNDAMENTAL MATRIX AT T =',F10.4/)
      CALL MATPR(N,N,FM,WORK1,LUNP)
  580 CONTINUE
C     CALCULATE H(T) ACCORDING TO THE TRAPEZOIDAL RULE.
      IF(K1) 590,590,595
  590 DO 591 I=1,NN
      FMR(I)=FM(I)/2.0
  591 CONTINUE
```

```
      C
      GO TO 605
  595 IF(K1-NPS) 600,610,610
  600 CONTINUE
      CALL GMADD(FMR,FM,FMR,N,N)
  605 K1=K1+1
      GO TO 400
  610 CONTINUE
      CALL LINK (WAF6)
      END
```

```
        C
C
C                                    THIS IS LINK 6 OF PART B
C
      EXTERNAL WAF7

      DIMENSION B(200),H(200),FM(100),FMR(100)
      DIMENSION WORK1(100),WORK8(20)
      DIMENSION IDUM1(80),RDUM1(45),MBLOK(50),RBLOK(40)
     1,RTEMP(300)
      DIMENSION IW(10),RW(40)

      COMMON IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW,FM,H

      EQUIVALENCE (IDUM1(1),LUNR)
      EQUIVALENCE (IDUM1(19),NUPOP)
      EQUIVALENCE (IDUM1(2),LUNP),(IDUM1(5),N),(IDUM1(9),N2)
      EQUIVALENCE (IDUM1(15),NPRTY)
      EQUIVALENCE (FMR(1),H(1))
      EQUIVALENCE (RTEMP(1),B(1))
      EQUIVALENCE (RDUM1(4),DELTH)

      DEFINE FILE 40(95,10,U,NEXT)
      DEFINE FILE 91(100,2,U,NXT91)
      DEFINE FILE92(200,2,U,NXT92)

      NN=N*N
      NN2=N*N2

      IF(3-NPRTY) 620,620,630
  620 CONTINUE
      WRITE(LUNP,621)
  621 FORMAT(1H ,/T14,'COEFFICIENT MATRICES OF THE STATE'
     1/T22,'DIFFEREN
     1E EQUATION'//T25,'MATRIX FM(T)'/)
      CALL MATPR(N,N,FM,WORK8,LUNP)
  630 CONTINUE

C     ADD HALF OF THIS LAST VALUE TO THE SUM FMR

      DO 500 I=1,NN
      WORK1(I)=FM(I)/2.0
  500 CONTINUE

      CALL GMADD(FMR,WORK1,FMR,N,N)

C     MULTIPLY THE SUM BY THE TIME SUBINTERVAL.

      CALL MSMUL(N,N,FMR,DELTH,FMR)
```

```
      C

C
      IF(10-NPRTY) 631,631,635
  631 WRITE(LUNP,632)
  632 FORMAT(1H0,'INTEGRAL OF FI(T)*DT OVER THE FIRST TIME
     1 INTERVAL COM
     1UTED ACCORDING TO THE TRAPEZOIDAL RULE'/)
      CALL MATPR(N,N,FMR,WORK8,LUNP)
  635 CONTINUE

C     COMPLETE THE CALCULATION OF H(T) =FMR*B.
C     FIRST READ MATRIX B.

      READ(92'1)B
      CALL GMPRD(FMR,B,WORK1,N,N,N2)

C     TRANSFER WORK1 TO H

      DO 655 I=1,NN2
      H(I)=WORK1(I)
  655 CONTINUE
      IF(3-NPRTY) 640,640,650
  640 WRITE(LUNP,641)
  641 FORMAT(1H ,/T25,'MATRIX H(T)'/)
      CALL MATPR(N,N2,H,WORK8,LUNP)
  650 CONTINUE

C     PUNCH COEFFICIENT MATRICES FM AND H ONTO CARDS IF
C     NUPOP IS 10

      IF(NUPOP-10) 690,651,690
  651 CONTINUE

C     A DECK OF BLANK CARDS MUST FOLLOW THE LAST DATA CARD
C     THIS IS A DUMMY READ STATEMENT

      READ(LUNR,670)
      WRITE(LUNR,660) (FM(I),I=1,NN)
  660 FORMAT(1H ,'FUNDAMENTAL MATRIX GIVEN COLUMNWISE IN
     1 FORMAT 5E15.6'
     1(5E15.6))
      READ(LUNR,670)
      WRITE(LUNR,670) (H(I),I=1,NN2)
  670 FORMAT(1H ,'COEFFICIENT MATRIX H GIVEN COLUMNWISE IN
     1 FORMAT 5E15.
     1'/(5E15.6))
  690 CALL LINK(WAF7)
      END
```

```
      C
C
C                    THIS IS LINK 7 OF PART B
C

      EXTERNAL WAF8,WAF9

      DIMENSION WORK1(10),WORK2(15),WORK3(15),WORK4(10)
     1,IWORK(20)
      DIMENSION C(150),D(300),H(200),FM(100)
      DIMENSION OW(200),YORXV(100),OV(15),Y(15)
      DIMENSION NOP(10),NCODE(10),NY(15),VD1(10),VD2(10)
     1,X(10)
      DIMENSION MV(5),MV1(5),NU(20),NU1(20),UD(20),UD1(20)
      DIMENSION IDUM1(80),RDUM1(45),MBLOK(50),RBLOK(40)
     1,RTEMP(300)
      DIMENSION IW(10),RW(40)
      DIMENSION NBLNK(15)

      COMMON IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW,FM,H

      EQUIVALENCE (IDUM1(4),LUNO)
      EQUIVALENCE (IDUM1(21),IDISC)
      EQUIVALENCE (IDUM1(2),LUNP),(IDUM1(5),N),(IDUM1(9),N2)
      EQUIVALENCE (IDUM1(6),M),(IDUM1(8),N1),(IDUM1(10),N3)
      EQUIVALENCE (IDUM1(11),N4),(IDUM1(12),N5)
      EQUIVALENCE   (IDUM1(13),NTD),(IDUM1(14),NPTS)
     1,(IDUM1(15),NPRTY)
      EQUIVALENCE (IDUM1(46),NY(1)),(IDUM1(61),NCODE(1))
     1,(IDUM1(71),NOP
     11))
      EQUIVALENCE (RDUM1(2),TO),(RDUM1(3),DELT)
      EQUIVALENCE (RDUM1(11),VD1(1))
      EQUIVALENCE (RDUM1(21),VD2(1)),(RDUM1(31),X(1))
      EQUIVALENCE (RBLOK(1),UD(1)),(RBLOK(21),UD1(1))
      EQUIVALENCE (MBLOK(1),MV(1)),(MBLOK(6),MV1(1))
     1,(MBLOK(11),NU(1)),
     1(MBLOK(31),NU1(1))
      EQUIVALENCE (RTEMP(1),D(1))
      EQUIVALENCE (IW(5),K1)

      DATA NBLNK/'  ','  ','  ','  ','  ','  ','  ','
     1 ','  ','  '
     1'  ','  ','  ','  '/

      DEFINE FILE 40(95,10,U,NEXT)
      DEFINE FILE 91(100,2,U,NXT91)
      DEFINE FILE 92(200,2,U,NXT92)
      DEFINE FILE 93(150,2,U,NXT93)
      DEFINE FILE 94(300,2,U,NXT94)
```

```
          C
C
      DEFINE FILE 96(1120,2,U,NXT96)
C     EXIT IF NO EXTERNAL FORCING FUNCTION IS GIVEN
      IF(IDISC) 2,2,3
    2 WRITE(LUNO,18)
   18 FORMAT(1H ,/1X,'PROGRAM STOPS AS NO DISTURBANCE HAS
     1 BEEN ENTERED'
      CALL EXIT
    3 CONTINUE
C     GIVE LENGTH OF VECTOR OW CONTAINING DELAYED DATA.
      MAXOW=200
C     GIVE LENGTH OF VECTOR YORXV TO CONTAIN VALUES OF Y OR
C     X.IT IS USES TO BUILD FILE 96 OF MAXIMUM LENGTH MXF96
      MAXYX=100
      MXF96=1120
      WRITE(LUNP,5)
    5 FORMAT(1H ,///T21,'TIME DOMAIN RESPONSE'//)
      IF(NOP(7)-1) 8,4,4
    4 CONTINUE
      WRITE(LUNP,6) (NBLNK(I),NY(I),I=1,N5)
    6 FORMAT(T4,'     TIME',9(5X,A2,'Y(',I2,')'))
      GO TO 11
    8 CONTINUE
      WRITE(LUNP,9) (NBLNK(I),NY(I),I=1,N5)
    9 FORMAT(T4,'     TIME',9(5X,A2,'X(',I2,')'))
   11 CONTINUE
      WRITE(LUNP,1)
    1 FORMAT(1H )
C     SET X INTO WORKING VECTOR WORK1.
      DO 10 I=1,N
      WORK1(I)=X(I)
   10 CONTINUE
C     ZERO WORKING VECTOR WORK2.
      DO 20 I=1,M
      WORK2(I)=0.0
   20 CONTINUE
C     ZERO COUNTER IOWCT USED BY BLDFF.
      IOWCT=0
```

```
      C
C     INITIALIZE COUNTER USED IN FILE 96.

      KY=1

C     CALCULATE INTEGER NUMBER OF SETS CONTAINED IN YORXV
C     VECTOR.

      NSETS=MAXYX/N5
      MAX1=NSETS*N5

C     FIND INITIAL TIME VALUE.

      KK=TO/DELT+.5
      T=DELT*FLOAT(KK)

C     START TIME DOMAIN COUNTER.

      K1=0

C     FIND NUMBER OF TIME INTERVALS.IF NPTS IS ZERO CALCU-
C     LATION ENDS AT TIME = 1.0

      IF(NPTS) 30,30,40
   30 NT=1.0/DELT+.5
      GO TO 50
   40 NT=NPTS
   50 CONTINUE

C     CALCULATION OF TIME DOMAIN RESPONSE INVOLVES BOTH
C     MATRICES D AND C OR ONLY C ACCORDING TO THE VALUE OF
C     NOP7).

      IF(NOP(7)) 200,200,70
   70 IF(NOP(7)-1) 80,80,90
   80 CONTINUE

C     ONLY MATRIX C IS NEEDED.

      READ(93'1)C
      GO TO 130
   90 CONTINUE

C     BOTH MATRICES C AND D ARE NEEDED.

      READ(93'1)C
      READ(94'1)D
   95 CONTINUE
      ND=NTD
   96 IF(N3-(1+2*ND)) 97,99,99
   97 ND=ND-1
```

```
      C

    GO TO 96
 99 CONTINUE
    CALL BLDFF(ND,TO,DELT,WORK1,N4,NU1,UD1,OV,N3,MV1,NCODE
   1,VD1,VD2,
   1MAXOW,OW,IWORK,IOWCT,K1,LUNP)
    IF(10-NPRTY) 100,100,120
100 WRITE(LUNP,101) T,(OV(I),I=1,N4)
101 FORMAT(1H0,10X,'V(',F6.3,')'/(10X,E15.6))
120 CONTINUE
    CALL GMPRD(D,OV,WORK2,M,N4,1)
130 CONTINUE
    CALL GMPRD(C,WORK1,WORK3,M,N,1)
    CALL GMADD(WORK3,WORK2,Y,M,1)
    IF(10-NPRTY) 140,140,200
140 WRITE(LUNP,141) T,(Y(I),I=1,M)
141 FORMAT(1H0,10X,'Y(',F6.3,')'/(10X,E15.6))
200 CONTINUE
    ND=NTD
201 IF(N1-(1+2*ND)) 202,209,209
202 ND=ND-1
    GO TO 201
209 CONTINUE
    CALL BLDFF(ND,TO,DELT,WORK1,N2,NU,UD,OV,N1,MV,NCODE
   1,VD1,VD2,
   1MAXOW,OW,IWORK,IOWCT,K1,LUNP)
    IF(10-NPRTY) 210,210,240
210 WRITE(LUNP,211) T,(WORK1(I),I=1,N)
211 FORMAT(1H0,10X,'X(',F6.3,')'/(10X,E15.6))
    WRITE(LUNP,231) T,(OV(I),I=1,N2)
231 FORMAT(1H0,10X,'U(',F6.3,')'/(10X,E15.6))
240 CONTINUE
    IF(3-NPRTY) 241,241,245
241 ISEE=99
    GO TO 250

C     DECIDE ABOUT OPTION NOP(8).

245 KQ=NOP(8)
    IF(KQ) 500,500,246
246 GO TO(248,300,300,350,350),KQ
248 IF(ISEE-99) 250,400,250
250 CONTINUE
    IF(NOP(7)-1) 270,260,260
260 DO 261 I=1,N5
    KP=NY(I)
    WORK3(I)=Y(KP)
261 CONTINUE
262 WRITE(LUNP,263) T,(WORK3(I),I=1,N5)
263 FORMAT(1H ,10F12.4)
    GO TO 280
```

```
      C
  270 DO 271 I=1,N5
      KP=NY(I)
      WORK3(I)=WORK1(KP)
  271 CONTINUE
      GO TO 262
  280 IF(ISEE-99) 400,245,400
  300 CONTINUE
C     VARIABLES OF INTEREST ARE STORED INTO DISK FILES TO BE
C     PLOTTED
C     ON THE SCOPE NOP(8)=2 OR ON THE PLOTTER NOP(8)=3
      DO 345 I=1,N5
      KR=I+1-KY+K1*N5
      KK=NY(I)
      IF(NOP(7)-1) 316,320,320
  316 YORXV(KR)=WORK1(KK)
      GO TO 329
  320 YORXV(KR)=Y(KK)
  329 IF(KR-MAX1) 345,330,330
  330 WRITE(96'KY) (YORXV(J),J=1,MAX1)
      KY=KY+MAX1
      IF(KY-MXF96) 345,345,335
  335 WRITE(LUNO,336)
  336 FORMAT(1H ,'TOO MANY VALUES IN FILE NO. 96'/)
      CALL EXIT
  345 CONTINUE
      IF(K1-NT) 400,348,348
  348 WRITE(96'KY) (YORXV(J),J=1,MAX1)
      CALL LINK(WAF10)
  350 WRITE(LUNO,351)
  351 FORMAT(1H ,'REQUESTED METHOD OF HANDLING OUTPUT
     1 VARIABLES NOT IMP
     1EMENTED'/)
      CALL EXIT
  400 CONTINUE
      ISEE=0
C     COMPUTE A NEW STATE VARIABLE VECTOR.
      CALL GMPRD(H,OV,WORK3,N,N2,1)
      CALL GMPRD(FM,WORK1,WORK4,N,N,1)
      CALL GMADD(WORK3,WORK4,WORK1,N,1)
C     STEP COUNTER.
      K1=K1+1
C     FIND NEW TIME VALUE.
```

```
      C
      KK=TO/DELT+.5
      KK=KK+K1
      T=DELT*FLOAT(KK)
C     CHECK FOR NUMBER OF ITERATIONS.
      IF(K1-NT) 420,420,500
C     DECIDE WHERE TO REENTER.
  420 IF(NOP(7)) 200,200,425
  425 IF(NOP(7)-1) 130,130,95
  500 CONTINUE
      IF(NOP(9)) 600,600,650
C     NO CHANGE DESIRED.
  600 NJOBS=NJOBS-1
      IF(NJOBS) 610,610,620
  610 CALL EXIT
  620 CALL LINK(LAST1)
  650 CALL LINK (WAF8)
      END
```

```
      C
C
C                                 THIS IS LINK 8 OF PART B
C

      EXTERNAL WAF7,WAF5,LAST1

      DIMENSION NOP(10),NCODE(10),VD1(10),VD2(10),X(10)
     1,NY(15)
      DIMENSION IV(10),RV(10),IALPH(1)
      DIMENSION IDUM1(80),RDUM1(45),MBLOK(50),RBLOK(40)
     1,RTEMP(300)
      DIMENSION IW(10),RW(40),H(200),FM(100)

      COMMON IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW,FM,H

      EQUIVALENCE (IDUM1(3),LUNI),(IDUM1(4),LUNO),(IDUM1(14)
     1,NPTS)
      EQUIVALENCE (IDUM1(15),NPRTY)
      EQUIVALENCE (IDUM1(16),IFLG2),(IDUM1(17),IFDFG)
     1,(IDUM1(46),NY(1))
      EQUIVALENCE (IDUM1(61),NCODE(1)),(IDUM1(71),NOP(1))
      EQUIVALENCE (RDUM1(4),DELTH),(RDUM1(2),TO),(RDUM1(3)
     1,DELT)
      EQUIVALENCE (RDUM1(11),VD1(1)),(RDUM1(21),VD2(1))
     1,(RDUM1(31),X(1)
      EQUIVALENCE (IDUM1(12),N5)
      EQUIVALENCE (IDUM1(20),INO)

      K=NOP(9)
      GO TO(5,200,300),K
    5 CONTINUE
      WRITE(LUNO,8)
    8 FORMAT(1H ,'ENTER   999,DIST.NO+100,DIST.CODE NO.,1ST
     1 DIST.VALUE,
     1ND DIST.VALUE **'/)

      CALL TWAIT

      NR=-1
      CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
      KK=IV(2)-100
      NCODE(KK)=IV(3)
      VD1(KK)=RV(1)
      VD2(KK)=RV(2)
      WRITE(LUNO,12)
   12 FORMAT(1H ,'IF A CHANGE OF ANY OF THE FOLLOWING IS
     1 DESIRED ENTER'
     1/'   ID NO.S AND NEW VALUES **' /'   999 ** OTHERWISE
     1 OR AS LAST
```

```
      C
   2NTRY'/)
    WRITE(LUNO,13)
 13 FORMAT(T1 ,'   1  WAY OF HANDLING THE OUTPUT...NOP(8)'
   1/'   2  INI
   1IAL TIME VALUE...............TO'/'   3  NO. OF TIME
   1 INTERVALS....
   2.....NPTS'/'   4  INITIAL STATE VECTOR...........X(0)'
   1/'   5  OUT
   3UT VARIABLE VECTOR...........NY'/'   6  FURTHER
   1 CHANGES..........
   4...NOP(9)'//)

    CALL TWAIT

 14 NR=-1
    CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
    IF(IV(1)-999) 30,21,21
 21 CALL LINK(WAF7)

 30 CONTINUE
    KK=IV(1)
    GO TO (35,40,45,50,55,60),KK
 35 NOP(8)=IV(2)
    GO TO 14
 40 TO=RV(1)
    GO TO 14
 45 NPTS=IV(2)
    GO TO 14
 50 DO 52 I=1,NR
    X(I)=RV(I)
 52 CONTINUE
    GO TO 14
 55 DO 57 I=2,NI
    NY(I-1)=IV(I)
 57 CONTINUE
    N5=NI-1
    GO TO 14
 60 NOP(9)=IV(2)
    GO TO 14

200 CONTINUE
    WRITE(LUNO,210)
210 FORMAT(1H ,'ENTER ID NO.S AND NEW VALUES **'/'
   1 999 ** AS LA
   1T ENTRY'/)
    WRITE(LUNO,220)
220 FORMAT(T1,'   1  OPTION(I) NO.,OPTION(I) VALUE....**'
   1/'   2  TO,D
   1LT,DELTH....................**'/'   3  NUMBER OF
   1 POINTS.........
```

```
      C
   2•••••••**'/'    4  OUTPUT VARIABLE VECTOR•••••••••••**'
   1/'    5  PRIO
   3ITY NUMBER FOR OUTPUT•••••••**'//)
    CALL TWAIT
230 NR=-1
    CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
    IF(IV(1)-999) 240,235,235
235 CALL LINK(WAF2)
240 CONTINUE
    KK=IV(1)
    GO TO(245,260,265,270,275),KK
245 DO 258 K=2,NI,2
    KK=IV(K)
    GO TO(246,247,248,249,250,251,252,253,254,255),KK
246 NOP(1)=IV(K+1)
    GO TO 258
247 NOP(2)=IV(K+1)
    GO TO 258
248 NOP(3)=IV(K+1)
    GO TO 258
249 NOP(4)=IV(K+1)
    GO TO 258
250 NOP(5)=IV(K+1)
    GO TO 258
251 NOP(6)=IV(K+1)
    GO TO 258
252 NOP(7)=IV(K+1)
    GO TO 258
253 NOP(8)=IV(K+1)
    GO TO 258
254 NOP(9)=IV(K+1)
    GO TO 258
255 NOP(10)=IV(K+1)
258 CONTINUE
    GO TO 230
260 TO=RV(1)
    DELT=RV(2)
    DELTH=RV(3)
    GO TO 230
265 NPTS=IV(2)
    GO TO 230
270 DO 271 I=2,NI
    NY(I-1)=IV(I)
271 CONTINUE
    GO TO 230
275 NPRTY=IV(2)
    GO TO 230
```

```
      C
300 CONTINUE
    WRITE(LUNO,310)
310 FORMAT(1H ,'ENTER T.F.TYPE OR PARAMETER **'/1X,'ENTER
   1 NOP(9) VALU
   1 TO STOP OR MAKE A DIFFERENT CHANGE'/)
    IFLG2=99
    IFDFG=1
    INO=INO+1

    CALL TWAIT

    CALL LINK(LAST1)
    END
```

```
      C
C
C                        THIS IS LINK 10 OF PART B
C

      EXTERNAL LAST1,WAF9,WAF8

      DIMENSION LABYX(10),LABYY(10),LABXX(10),IVEC1(20)
     1,IVECT(20)
      DIMENSION LABY(10),LABX(10),ISAVE(15),IPTNY(10)
      DIMENSION Y(150)
      DIMENSION NOP(10),NY(15),IV(10),RV(10),IALPH(40)
      DIMENSION IDUM1(80),RDUM1(45),MBLOK(50),RBLOK(40)
     1,RTEMP(300)
      DIMENSION IW(10),RW(40),H(200),FM(100)
      DIMENSION RADD(5)

      COMMON IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW,FM,H

      EQUIVALENCE (IDUM1(2),LUNP),(IDUM1(4),LUNO),(IDUM1(12)
     1,N5)
      EQUIVALENCE (IDUM1(15),NPRTY),(IDUM1(41),NJOBS)
     1,(IDUM1(46),NY(1))
      EQUIVALENCE (IDUM1(71),NOP(1)),(IDUM1(3),LUNI)
      EQUIVALENCE (IW(5),K1)
      EQUIVALENCE (RDUM1(2),TO),(RDUM1(3),DELT)
      EQUIVALENCE (RDUM1(1),RADD(1))
      EQUIVALENCE (RTEMP(151),Y(1))

      DATA LABYY/'  ','  ','  ','  ','  ',' Y','( ','  ',')
     1 ','  '/
      DATA LABYX/'  ','  ','  ','  ','  ',' X','( ','  ',')
     1 ','  '/
      DATA LABXX/'  ','  ','  ','T/','TM','AX','  ','  ','
     1 ','  '/
      DATA IVEC1/'  ','  ','  ','  ','TI','ME',' D','OM'
     1,'AI','N ','RE'
     1'SP','ON','SE','  ','  ','  ','  ','  ','  '/
      DATA ISAVE/'1 ','2 ','3 ','4 ','5 ','6 ','7 ','8 ','9
     1 ','10','11'
     1'12','13','14','15'/

      DEFINE FILE 40(95,10,U,NEXT)
      DEFINE FILE 91(100,2,U,NXT91)
      DEFINE FILE 92(200,2,U,NXT92)
      DEFINE FILE 93(150,2,U,NXT93)
      DEFINE FILE 94(300,2,U,NXT94)
      DEFINE FILE 96(1120,2,U,NXT96)
      DEFINE FILE 99(160,10,U,NXT99)
      DEFINE FILE 100(1120,2,U,NX100)
```

```
      C
      IRET=IDUM1(51)
      IF(IRET-99) 200,250,200
  200 CONTINUE
      WRITE(99'1)(IDUM1(I),I=1,80),(RDUM1(I),I=1,45)
     1,(MBLOK(I),I=1,50),
     1(RBLOK(I),I=1,40),(RTEMP(I),I=1,300),(IW(I),I=1,10)
     1,(RW(I),I=1,40
     2,(H(I),I=1,200),(FM(I),I=1,100)
C     INITIALIZE COUNTERS
      IPTCT=0
      KY=0
C     SET RETURN FLAG
      IRET=99
      GO TO 300
  250 CONTINUE
C     SAVE VALUES FROM COMMON BEFORE RESTORING THE OLD ONE
      IPTAL=IDUM1(52)
      IPTCT=IDUM1(53)
      KY=IDUM1(54)
      READ(99'1) IDUM1,RDUM1,MBLOK,RBLOK,RTEMP,IW,RW,H,FM
  300 CONTINUE
      IF(IPTAL-99) 508,3,508
  508 CONTINUE
  509 WRITE(LUNO,510)
  510 FORMAT(1H ,'ENTER 1 ** IF PLOT IS DESIRED'/'        0 *
     1* OTHERWISE
     1/)
      CALL TWAIT
      NR=-1
      CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
      IF(IV(1)) 520,520,550
  520 WRITE(LUNO,525)
  525 FORMAT(1H ,'ENTER NOP(9) VALUE FOR FURTHER CHANGES'/'
     1      0 **
     1THERWISE'/)
      CALL TWAIT
      NR=-1
      CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
      NOP(9)=IV(1)
      IF(NOP(9)) 530,530,535
  530 NJOBS=NJOBS-1
      IF(NJOBS) 532,532,533
  532 CALL EXIT
```

```
      C
533 CALL LINK(LAST1)
535 CALL LINK(WAF8)
550 CONTINUE
    WRITE(LUNO,551)
551 FORMAT(1H ,'ENTER 1 ** FOR PLOTTING ALL YS IN THE
   1 STANDARD FORM'/
   1      0 ** OTHERWISE'/)
    CALL TWAIT
    NR=-1
    CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
    IF(IV(1)) 570,570,560
560 IPTAL=99
    GO TO 3
570 IPTAL=0
    WRITE(LUNO,571)
571 FORMAT(1H ,'ENTER ID NO. FOR DESIRED VARIABLE Y'//'
   1       ID
   1     Y'/)
    DO 574 K=1,N5
    WRITE(LUNO,573) K,NY(K)
573 FORMAT(1H ,2I10)
574 CONTINUE
    CALL TWAIT
    NR=-1
    CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
    IF(IV(1)) 509,509,580
580 IPTSY=99
    KY=IV(1)
    WRITE(LUNO,590)
590 FORMAT(1H ,'ENTER 1 ** FOR STANDARD GRAPH'/'       0 *
   1* OTHERWISE
   1/)
    CALL TWAIT
    NR=-1
    CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
    IF(IV(1)) 600,600,3
600 WRITE(LUNO,610)
610 FORMAT(1H ,'ENTER ID NO. AND NEW VALUES **'//'  1
   1 YMIN,YMAX.....
   1...........................**'/'  2  PLOT
   1 STATUS..1=NEW,2=OVERTO
   2,3=OVERBOTTOM..**'/'  3  PLOT TYPE..-VE=LINE,0=+,1=X
   1 ETC.........
   3..**'/'  4  DEVICE..1=SCOPE,2=PLOTT
   1ER..................**'/'  5
   4LOT POSITION..1=TOP,2=BOTTOM..............**')
    WRITE(LUNO,615)
615 FORMAT(1H ,' 6 **..LABEL ON X AXIS..............
   1*..(20A1)..*'/'
   17 **..LABEL ON Y AXIS...............*..(20A1)..*'/'  8
```

```
      C
     1 **..TITLE.
     2.......................*..(40A1)..*'//1X,'LAST ENTRY
     1 IS 999 **'/)
      CALL TWAIT
      DO 505 I=1,10
      IPTNY(I)=0
  505 CONTINUE
  650 NR=-1
      CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
      IF(IV(1)-999) 670,3,3
  670 KK=IV(1)
      GO TO (710,720,730,740,750,760,770,780),KK
  710 YMIN=RV(1)
      YMAX=RV(2)
      IPTNY(1)=1
      GO TO 650
  720 IU=IV(2)
      IPTNY(2)=1
      GO TO 650
  730 IP=IV(2)
      IPTNY(3)=1
      GO TO 650
  740 ID=IV(2)
      IPTNY(4)=1
      GO TO 650
  750 IG=IV(2)
      IPTNY(5)=1
      GO TO 650
  760 CONTINUE
      NI=0
      NR=0
      NA=20
      CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
      CALL AFCRT(NA,IALPH,L,LABX)
      IPTNY(6)=1
      GO TO 650
  770 CONTINUE
      NI=0
      NR=0
      NA=20
      CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
      CALL AFCRT(NA,IALPH,L,LABY)
      IPTNY(7)=1
      GO TO 650
  780 CONTINUE
      NI=0
      NR=0
      NA=40
      CALL FFINP(IV,RV,IALPH,NI,NR,NA,LUNO,LUNI)
      CALL AFCRT(NA,IALPH,L,IVECT)
```

```
      C
      IPTNY(8)=1
      GO TO 650
    3 CONTINUE
      ICT=1
      KFL=1
      MAXYV=150
      MX100=1120
      MXTMP=150
      NSETS=MXTMP/N5
      NVAL=NSETS*N5
      KI=TO/DELT+.5
      KF=KI+K1
      XMIN=DELT*FLOAT(KI)
      XMAX=DELT*FLOAT(KF)
      KN=K1+1
      NN5=KN*N5
      NVALC=NVAL
      IF(IPTSY-99) 34,40,34
   34 IF(IPTAL-99) 40,35,40
   35 IF(KY-N5) 36,509,509
   36 KY=KY+1
   40 CONTINUE
   50 IF(NN5-NVALC) 60,60,70
   60 KK=NN5-KFL+1
      READ(96'KFL) (RTEMP(I),I=1,KK)
      GO TO 100
   70 READ(96'KFL) (RTEMP(I),I=1,NVAL)
C     SORT OUT VALUES FROM RTEMP INTO WORKING VECTOR Y.
      DO 75 K=1,NSETS
      KT=KY+N5*(K-1)
      Y(K)=RTEMP(KT)
   75 CONTINUE
C     CHECK FOR ROOM IN FILE 100
      IF(ICT+NSETS-MX100) 72,72,80
   72 WRITE(100'ICT) (Y(J),J=1,NSETS)
C     STEP CONTERS.
      ICT=ICT+NSETS
      KFL=KFL+NVAL
      NVALC=NVALC+NVAL
      GO TO 50
   80 WRITE(LUNO,81)
   81 FORMAT(1H ,'TOO MANY VALUES IN FILE 100'/)
      CALL EXIT
```

```
      C
  100 CONTINUE
      K2=KK/N5+0.5
      DO 120 K=1,K2
      KT=KY+N5*(K-1)
      Y(K)=RTEMP(KT)
  120 CONTINUE
C     CHECK FOR ROOM IN FILE 100
      IF(ICT+K2-MX100) 119,119,80
  119 WRITE(100'ICT) (Y(J),J=1,K2)
      KT=NY(KY)
      IF(10-NPRTY) 123,123,128
  123 DO 126 I=1,KN
      READ(100'I) TEST
      WRITE(LUNP,124) KT,TEST
  124 FORMAT(1H ,5X,'Y(',I2,')'//(E15.6))
  126 CONTINUE
  128 CONTINUE
      KLAB=ISAVE(KT)
      IF(IPTNY(5)) 140,140,150
  140 IPTCT=IPTCT+1
  141 IF(IPTCT-2) 142,142,143
  142 IG=IPTCT
      GO TO 150
  143 IPTCT=IPTCT-2
      GO TO 141
  150 CONTINUE
      IF(IPTNY(7)) 5,5,10
    5 IF(NOP(7)) 6,6,8
    6 DO 7 I=1,10
      LABY(I)=LABYX(I)
    7 CONTINUE
      LABY(8)=KLAB
      GO TO 10
    8 DO 9 I=1,10
      LABY(I)=LABYY(I)
    9 CONTINUE
      LABY(8)=KLAB
   10 CONTINUE
      IF(IPTNY(6)) 11,11,13
   11 DO 12 I=1,10
      LABX(I)=LABXX(I)
   12 CONTINUE
   13 IF(IPTNY(8)) 14,14,16
   14 DO 15 I=1,20
      IVECT(I)=IVEC1(I)
   15 CONTINUE
   16 CONTINUE
```

```
      C

C
      IF(IPTNY(3)) 18,18,20
   18 IP=-1
   20 CONTINUE
      IF(IPTNY(2)) 22,22,25
   22 IU=1
   25 CONTINUE
      IF(IPTNY(4)) 28,28,30
   28 ID=NOP(8)-1
   30 CONTINUE

C     BUILD NEW COMMON TO BE USED IN LINK 9.

      DO 155 I=1,20
      IDUM1(I)=IVECT(I)
  155 CONTINUE
      DO 160 I=1,10
      K=I+20
      IDUM1(K)=LABX(I)
  160 CONTINUE
      DO 165 I=1,10
      K=I+30
      IDUM1(K)=LABY(I)
  165 CONTINUE
      DO 170 I=1,10
      K=I+40
      IDUM1(K)=IPTNY(I)
  170 CONTINUE
      IDUM1(51)=IRET
      IDUM1(52)=IPTAL
      IDUM1(53)=IPTCT
      IDUM1(54)=KY
      IDUM1(55)=ID
      IDUM1(56)=IG
      IDUM1(57)=IU
      IDUM1(58)=IP
      IDUM1(59)=KN
      RADD(1)=XMAX
      RADD(2)=XMIN
      RADD(3)=YMAX
      RADD(4)=YMIN
      CALL LINK(WAF9)
      END
```

```
      C
C
C                                  THIS IS LINK 9 OF PART B
C
      EXTERNAL WAF10

      DIMENSION LABY(10),LABX(10),IPTNY(10),IVECT(20)
      DIMENSION XW(50),YW(50)

      COMMON IVECT,LABX,LABY,IPTNY,IRET,IPTAL,IPTCT,KY,ID,IG
     1,IU,IP,KN
      COMMON IDUM9(21)
      COMMON XMAX,XMIN,YMAX,YMIN

      DEFINE FILE 40(95,10,U,NEXT)
      DEFINE FILE 91(100,2,U,NXT91)
      DEFINE FILE 92(200,2,U,NXT92)
      DEFINE FILE 93(150,2,U,NXT93)
      DEFINE FILE 94(300,2,U,NXT94)
      DEFINE FILE 96(1120,2,U,NXT96)
      DEFINE FILE 99(160,10,U,NXT99)
      DEFINE FILE 100(1120,2,U,NX100)
      DEFINE FILE 101(1120,2,U,NX101)

      LUNPS=7
      KK=1
      IS=2
      IL=1
      IT=2
      IB=1
      IR=2
      LO=100
      L1=101
      II=0

      IF(IPTNY(1)) 160,160,180

  160 CALL NWMAX(YW,KK,YMIN,YMAX,IR,II,LO,KN)

  180 CALL NWPLT(XW,YW,KK,ID,IS,IG,IL,IT,IB,IU,IR,IP,XMAX
     1,XMIN,NCYCX,
     1YMAX,YMIN,NCYCY,LABX,LABY,L1,LO,KN)

      IF(IU-1) 190,190,280
  190 CONTINUE
      CALL SCALF(1.,1.,8.5,0.0)
      IF(IG-1) 200,200,220
  200 CALL FPLOT(+1,+1.5,+9.75)
      CALL FCHAR(+1.5,+9.75,0.15,0.20,+0.0)
```

```
      C
      GO TO 250
  220 CALL FPLOT(+1,+1.5,+4.75)
      CALL FCHAR(+1.5,+4.75,0.15,0.20,+0.0)
  250 CONTINUE
      WRITE(LUNPS,253) IVECT
  253 FORMAT(20A2)
      CALL FPLOT(+1,+8.5,+0.0)
  280 CONTINUE
      CALL LINK(WAF10)
      END
```

APPENDIX F

Listing of Subroutines

GMADD,GMPRD,GMSUB,MINV,PMPY,PRBM,PQFB,PDER,SIMQ: are given in reference 13.

FCHAR,FPLOT,NWMAX,NWPLT,SCALF: see DACS Centre.

```
      C
C
      SUBROUTINE FFIOR(IV,RV,IALPH,NI,NR,NA,ID1,ID2)

C     PURPOSE
C          TO READ AND WRITE DATA IN FREE FORMAT
C          SUBROUTINE FFINP IS ANALOGOUS BUT LIMITED TO
C          READING ONLY
C
C     PARAMETERS
C          IV = VECTOR CONTAINING NI INTEGERS IN THE SAME
C               ORDER AS SUPPLIED WITH THE INPUT LIST
C          RV = VECTOR CONTANING NR REAL VALUES
C          IALPH = VECTOR CONTAINING NA ALPHANUMERIC VALUES
C          NI,NR,NA = NUMBER OF INTEGERS,REALS,ALPHAMERICS
C          ID1 = LOGICAL UNIT NUMBER FOR OUTPUT DEVICE
C          ID2 = LOGICAL UNIT NUMBER FOR INPUT DEVICE
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          NONE
C

      DIMENSION IV(1),RV(1),IALPH(1)
      DIMENSION ICODE(17),INPUT(81)
      DATA ICODE/'  ','0 ','1 ','2 ','3 ','4 ','5 ','6 ',
     1'7 ','8 ','9 ','E ','+ ','- ',', ','. ','* '/
      DATA ICARD/0/
      IWRIT=ID1
      IREAD=ID2
      IICNT=1
      IRCNT=1
      IACNT=1
      INPUT(81)=ICODE(1)
  107 READ(IREAD,1000) (INPUT(I),I=1,80)
 1000 FORMAT(80A1)

      WRITE(IWRIT,1100) (INPUT(I),I=1,80)
 1100 FORMAT(1H ,80A1)

      ICARD=ICARD+1
      IVLD=0
      IEXP=0
      IVAL=0
      IDEC=0
      ICALC=0
      ISIGN=0
      IDCNT=1
      IMULT=1
      I=1
  120 CONTINUE
      IF(INPUT(I)-5440) 110,106,110
```

```
      C
  110 CONTINUE
      DO 101 J=1,17
      IF(ICODE(J)-INPUT(I)) 101,102,101
  101 CONTINUE
      GO TO 150
  102 GO TO (200,300,300,300,300,300,300,300,300,300,300,
     1400,500,500,700,800,600) , J
  200 IF(IVLD) 150,105,210
  210 CALC=1
      GO TO 711
  300 IVLD=1
      ICALC=0
      IF(IDEC) 150,301,302
  301 IVAL=IVAL*10+(J-2)
      GO TO 105
  302 RVAL=RVAL+(J-2.0)/10.0**IDCNT
      IDCNT=IDCNT+1
      GO TO 105
  400 IF(IVLD) 150,150,401
  401 IF(IEXP) 150,402,150
  402 IEXP=1
      IF(IDEC) 150,404,403
  403 RV(IRCNT)=RVAL*IMULT
      GO TO 405
  404 RV(IRCNT)=IVAL*IMULT
  405 IMULT=1
      IVLD=0
      ISIGN=0
      IDEC=0
      IVAL=0
      GO TO 105
  500 IF(ISIGN) 150,501,150
  501 IF(IVLD) 150,502,150
  502 ISIGN=1
      IF(J-13) 150,105,503
  503 IMULT=-1
      GO TO 105
  600 I=I+1
      IF(ICODE(17)-INPUT(I)) 601,602,601
  601 DO 605 K=I,80
      IF(ICODE(17)-INPUT(K+1)) 605,607,605
  605 CONTINUE
      GO TO 150
  607 IF(NA) 611,611,609
  609 DO 610 KK=I,K
      IALPH(IACNT)=INPUT(KK)
  610 IACNT=IACNT+1
  611 I=K+1
      GO TO 200
  602 IF(NR) 603,604,604
```

```
      C
  603 NR=IRCNT-1
      NI=IICNT-1
      NA=IACNT-1
      RETURN
  604 WRITE(IWRIT,1002) ICARD
 1002 FORMAT(1H ,'DATA END ENCOUNTERED WHILE READING CARD NO
     1 ',I3)
      RETURN
  700 IF(ICALC) 150,711,712
  712 ICALC=0
      GO TO 105
  711 IF(IEXP) 150,703,702
  702 RV(IRCNT)=RV(IRCNT)*10.0**(IVAL*IMULT)
      IRCNT=IRCNT+1
      GO TO 706
  703 IF(IDEC) 150,705,704
  704 RV(IRCNT)=RVAL*IMULT
      IRCNT=IRCNT+1
      GO TO 706
  705 IV(IICNT)=IVAL*IMULT
      IICNT=IICNT+1
  706 IDEC=0
      IEXP=0
      ISIGN=0
      IVAL=0
      IVLD=0
      IMULT=1
      IDCNT=1
      GO TO 105
  800 IF(IEXP) 150,801,150
  801 IF(IDEC) 150,802,150
  802 IVLD=1
      IDEC=1
      ICALC=0
      RVAL=IVAL
  105 I=I+1
      IF(I-81) 120,120,106
  106 IF(NR) 107,108,108
  108 IF(NI-IICNT+1) 10,20,107
   10 WRITE(IWRIT,1003) ICARD
 1003 FORMAT(1H ,'TOO MANY NUMERICAL VALUES- CARD NO ',I3)
   40 RETURN
   20 IF(NR-IRCNT+1) 10,60,107
   60 IF(NA-IACNT+1) 10,40,107
  150 WRITE(IWRIT,1001) ICARD
 1001 FORMAT(1H ,'ERROR IN CARD ',I3)
      RETURN
      END
```

```
      C
C
      SUBROUTINE FIND(ITT,INOF,NBLKV,ITFV,NIV,NRV,IV,RV,NIF
     1,NRF,IVF,RVF
     1NIOF,NRIOF,IIOF,RVIOF)

C     PURPOSE
C          TO SORT A SET OF TRANSFER FUNCTION PARAMETERS
C          AND I/O PARAMETERS OUT OF VECTORS IV AND RV
C
C     PARAMETERS
C          ITT     = LOCATION WHERE THE SEARCH HAS TO START
C          INOF    = TOTAL NUMBER OF ENTRIES IN NBLKV VECTOR
C          NBLKV   = VECTOR CONTAINING INPUT ID NUMBERS FOR
C                    TF AND ASSOCIATED I/O
C          ITFV    = VECTOR CONTAINING TF (BLOCK) NUMBERS
C          NIV     = VECTOR CONTAINING NUMBER OF INTEGERS AS-
C                    SOCIATED WITH THE TF OR THE I/O DATA IN
C                    NBLKV VECTOR
C          NRV     = VECTOR SIMILAR TO NIV FOR REAL VALUES
C          IV      = INPUT VECTOR CONTAINING ALL INTEGER VAL-
C                    UES SUPPLIED VIA FFINP AND STORED IN
C                    COM-
C                    MON
C          RV      = VECTOR SIMILAR TO IV FOR REAL VALUES
C          NIF     = NUMBER OF INTEGERS SORTED OUT IN IVF
C          NRF     = NUMBER OF REAL VALUES SORTED OUT INTO
C          RVF
C          IVF     = VECTOR CONTAINING INTEGER VALUES RELATI-
C                    VE TO TF GIVEN IN ITFV VECTOR
C          RVF     = SIMILAR TO IVF FOR REAL VALUES
C          NIOF    = NUMBER OF CONFIGURATION INTEGER VALUES
C                    SORTED OUT INTO IIOF VECTOR
C          NRIOF   = SIMILAR TO IIOF FOR REAL VALUES  SORTED
C                    OUT INTO RVIOF VECTOR
C          IIOF    = VECTOR CONTAINING INTEGER VALUES OF CON-
C                    FIGURATION DATA ASSOCIATED WITH TF IN
C                    VECTOR ITFV
C          RVIOF   = VECTOR SIMILAR TO IIOF FOR REAL VALUES
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          NONE
C

      DIMENSION NBLKV(1),ITFV(1),NIV(1),NRV(1),IV(1),RV(1)
     1,IVF(1),RVF(1
     1,IIOF(1),RVIOF(1)

      NFLAG=1
      IT=ITT
```

```
      C
C     SUM ALL VALUES IN NIV,NRV UP TO THE LAST ONE.
    2 IT1=IT-1
      ITEM=0
      ITEM1=0
C     BYPASS THE SUMMING POINT IF IT IS ONE.
      IF(IT1-1) 9,7,7
    7 DO 8 K=1,IT1
      ITEM=ITEM+NIV(K)
      ITEM1=ITEM1+NRV(K)
    8 CONTINUE
    9 CONTINUE
      GO TO (11,12),NFLAG
   11 NIF=NIV(IT)
      GO TO 18
   12 NIOF=NIV(IT)
C     BYPASS THE SEARCH IF NI IS ZERO.
   18 IF(NIV(IT)) 50,50,20
   20 K1=ITEM+1
      K2=ITEM+NIV(IT)
      DO 30 KK=K1,K2
      KK1=KK-K1+1
      GO TO (28,29),NFLAG
   28 IVF(KK1)=IV(KK)
      GO TO 30
   29 IIOF(KK1)=IV(KK)
   30 CONTINUE
   50 GO TO (51,52),NFLAG
   51 NRF=NRV(IT)
      GO TO 53
   52 NRIOF=NRV(IT)
C     BYPASS THE SEARCH IF NR IS ZERO.
   53 IF(NRV(IT)) 64,64,54
   54 K1=ITEM1+1
      K2=ITEM1+NRV(IT)
      DO 60 KK=K1,K2
      KK1=KK-K1+1
      GO TO (58,59),NFLAG
   58 RVF(KK1)=RV(KK)
      GO TO 60
   59 RVIOF(KK1)=RV(KK)
   60 CONTINUE
```

```
      C

C     SEARCH FOR CORRESPONDING I/O DATA IN VECTOR NBLKV.

   64 IF(NFLAG-2) 65,120,120
   65 K1=ITFV(IT)+100
      DO 100 KK=1,INOF
      IF(NBLKV(KK)-K1) 100,70,100
   70 NFLAG=NFLAG+1
      IT=KK
      GO TO 2
  100 CONTINUE
  120 RETURN
      END
```

```
      C
C
      SUBROUTINE GTCC1(GAIN,DELAY,TLAG,NCD,I2,I3,RWO,WCC)
C     PURPOSE
C        TO PROVIDE CONTROLLER CONSTANTS FOR A FIRST ORDER
C        LAG ACCORDING TO TYPE OF CONTROLLER,CRITERIA
C        AND DISTURBANCE(SEE USER'S MANUAL FOR DETAILS)
C
C     PARAMETERS
C        GAIN = PROCESS GAIN
C        DELAY = TIME DELAY (ABSOLUTE VALUE)
C        TLAG = TIME LAG
C        NCD = CODE NUMBER FOR THE CONTROLLER
C              P=4,PI=5,PID=6
C        I2 = CODE FOR THE CRITERION (1 TO 8)
C        I3 = CODE FOR THE DISTURBANCE (1 OR 2)
C        RWO = OUTPUT VECTOR CONTAINING P,PI,PID CONSTANTS
C        WCC = VECTOR CONTAING NUMERICAL VALUES USED IN
C              CALCULATING THE CONTROLLER CONSTANTS
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C        NONE
C
      DIMENSION RWO(1),WCC(1)

      R=DELAY/TLAG
      RG=R*GAIN

      K=NCD-3
      GO TO (100,200,300),K

  100 GO TO(110,110,110,110,110,160,170,180),I2

  110 KK=(I2-1)*2+1
      A=WCC(KK)
      B=WCC(K+1)
      RWO(1)=(1.0/GAIN)*A*R**(-B)
      RETURN

  160 RWO(1)=(1.0/GAIN)*R**(-1.0)+0.333
      RETURN

  170 GO TO(175,175),I3
  175 RWO(1)=0.7/RG
      RETURN

  180 GO TO(185,185),I3
  185 RWO(1)=0.3/RG
      RETURN
```

```
      C

200 GO TO(210,210,210,210,210,260,270,280),I2

210 KK=(I2-1)*4+1
    A=WCC(KK+10)
    B=WCC(KK+11)
    C=WCC(KK+12)
    D=WCC(KK+13)

    RWO(1)=(1.0/GAIN)*A*R**(-B)
    RWO(2)=TLAG*C*R**D
    RETURN

260 RWO(1)=(1.0/GAIN)*(0.082+R**(-1.0))
    RWO(2)=(3.33*R+0.3*R**2)/(1.0+2.2*R)
    RETURN

270 GO TO(273,275),I3
273 RWO(1)=0.6/RG
    RWO(2)=RWO(1)*DELAY*GAIN/0.6
    RETURN
275 RWO(1)=0.7/RG
    RWO(2)=RWO(1)*DELAY*RG/0.3
    RETURN

280 GO TO (283,285),I3
283 RWO(1)=0.35/RG
    RWO(2)=RWO(1)*DELAY*GAIN/0.3
    RETURN
285 RWO(1)=0.6/RG
    RWO(2)=RWO(1)*DELAY*RG/0.15
    RETURN

300 GO TO(310,310,310,310,310,360,370,380),I2
310 KK=(I2-1)*6+1
    A=WCC(KK+30)
    B=WCC(KK+31)
    C=WCC(KK+32)
    D=WCC(KK+33)
    E=WCC(KK+34)
    F=WCC(KK+35)

    RWO(1)=(1.0/GAIN)*A*R**(-B)
    RWO(2)=TLAG*C*R**D
    RWO(3)=TLAG*E*R**F
    RETURN

360 RWO(1)=(1.0/GAIN)*(0.270+1.35*R**(-1.0))
    RWO(2)=TLAG*(2.5*R+0.5*R**2.0)/(1.0+0.6*R)
    RWO(3)=TLAG*R*0.37/(1.0+0.2*R)
```

```
      C
      RETURN
  370 GO TO(373,375),I3
  373 RWO(1)=0.95/RG
      RWO(2)=RWO(1)*DELAY*GAIN/0.7
      RWO(3)=0.45*DELAY/(RG*RWO(1))
      RETURN
  375 RWO(1)=1.2/RG
      RWO(2)=RWO(1)*DELAY*RG/0.6
      RWO(3)=0.5*DELAY/(RG*RWO(1))
      RETURN

  380 GO TO(383,385),I3
  383 RWO(1)=0.6/RG
      RWO(2)=RWO(1)*DELAY*GAIN/0.6
      RWO(3)=0.3*DELAY/(RG*RWO(1))
      RETURN
  385 RWO(1)=0.95/RG
      RWO(2)=RWO(1)*DELAY*RG/0.4
      RWO(3)=0.4*DELAY/(RG*RWO(1))
      RETURN
      END
```

```
      C
C
      SUBROUTINE TWAIT
C     PURPOSE
C          TO OUTPUT A MESSAGE ON THE 1816 WAITING FOR INPUT
C     NOTES
C          WRITTEN IN ASSEMBLER
C
                         ENT      TWAIT
                   TWAIT DC       *-*
                         LIBF     TYPEN
                         DC       /0005
                         MDX      *-3
                         BSC  I   TWAIT
                         END
      END
```

```
      C
C
      SUBROUTINE MATPR (L,L1,A,WA,LUNP)
C     PURPOSE
C          TO PRINT OUT A MATRIX ARRANGED COLUMNWISE
C          IN TWO DIMENSIONAL FORM
C
C     PARAMETERS
C          L = NUMBER OF ROWS IN MATRIX A
C          L1 = NUMBER OF COLUMNS
C          A = INPUT VECTOR OF LENGTH L*L1
C          WA = WORKING VECTOR OF LENGTH L1
C          LUNP = LOGICAL UNIT NUMBER FOR OUTPUT
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          NONE
C
C     NOTES
C          METHOD LIMITED TO LX6 TWO-DIMENSIONAL MATRIX
C
      DIMENSION A(1),WA(1)
      DO 2 I=1,L
      DO 1 J=1,L1
      K=I+(J-1)*L
      WA(J)=A(K)
    1 CONTINUE
      WRITE(LUNP,101) (WA(J),J=1,L1)
  101 FORMAT(1H0,10E12.4/)
    2 CONTINUE
      RETURN
      END
```

```
      C
C
      SUBROUTINE MATCO (TDV,NRW,NCL,BINP,N2,BOUT,N1,MV,NU
     1,UD)
C     PURPOSE
C          TO COMPRESS A SPARSE MATRIX TO A NON-ZERO COLUMN
C          ONE
C
C     PARAMETERS
C          TDV     = TIME DELAY ASSOCIATED WITH INPUT MATRIX
C          NRW     = NUMBER OF ROWS OF INPUT MATRIX BINP
C          NCL     = NUMBER OF COLUMNS OF BINP
C          BINP    = INPUT MATRIX
C          N2      = NUMBER OF COLUMNS OF OUTPUT MATRIX BOUT
C                    ALSO LENGTH OF VECTORS NU AND UD
C          BOUT    = OUTPUT MATRIX (NO ZERO COLUMN)
C          N1      = LENGTH OF VECTOR MV
C          MV      = VECTOR CONTAINING  N2 VALUES
C          NU      = VCTOR CONTAINING NON ZERO COLUMN NUMBERS
C          UD      = VECTOR CONTAINING VALUES OF TIME DELAY
C                    FOR EACH NON-ZERO COLUMN
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          NONE
C
C     NOTES
C          IF MORE THAN ONE MATRIX IS TO BE COMPRESSED N1
C          AND N2 MUST BE INITIALIZED TO ZERO AHEAD OF THE
C          FIRST ONE WHILE TDV HAS TO BE SET EVERY TIME IT
C          DIFFERS FROM ITS PREVIOUS VALUE
C
      DIMENSION BINP(1),BOUT(1),MV(1),NU(1),UD(1)
C     SET INITIAL VALUES.
      J=1
      NCCT=0
      NS=0
    5 J1=J+NRW-1
C     SEARCH FOR A NON ZERO COLUMN.
      NCCT=NCCT+1
      DO 20 I=J,J1
      IF(BINP(I)) 10,20,10
   10 N2=N2+1
      NS=NS+1
      NU(N2)=NCCT
      UD(N2)=TDV
```

```
      C

C     SET THIS COLUMN INTO BOUT.

      DO 15 K=J,J1
      K1=K-J+1+(N2-1)*NRW
      BOUT(K1)=BINP(K)
   15 CONTINUE

C     EXIT FROM THIS COLUMN.

      GO TO 30
   20 CONTINUE
   30 J=J+NRW
      IF(NCCT-NCL) 5,40,40
   40 N1=N1+1
      MV(N1)=NS
      RETURN
      END
```

```
     C
C
      SUBROUTINE CHEQN (N,A,WA,WB,WC,V,R,IER)
C     PURPOSE
C          TO DETERMINE THE COEFFICIENTS OF THE CHARACTERIS-
C          TIC EQUATION OF A GIVEN MATRIX BY USING KRYLOV'S
C          METHOD.(A.RALSTOW,''A FIRST COURSE IN NUMERICAL
C          ANALYSIS,MCGRAW-HILL,1965)
C
C     PARAMETERS
C          N = DIMENSION OF GIVEN N*N MATRIX A
C          A = GIVEN VECTOR OF DIMENSION N*N
C          WA = WORKING VECTOR OF DIMENSION N
C          WB = WORKING VECTOR OF DIMENSION N
C          WC = WORKING VECTOR OF DIMENSION N*N
C          V = WORKING VECTOR OF DIMENSION N INTERNALLY
C              GENERATED
C          R = RESULTING VECTOR OF COEFFICIENTS OF DIMENSION
C              N+1 ORDERED FROM LOW TO HIGH POWERS
C          IER = ERROR CODE =0 O.K.= 1 SINGULAR MATRIX FOUND
C              IN SOLVING SYSTEM OF SIMULTANEOUS EQUATIONS
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          GMPRD,SIMQ
C
      DIMENSION A(1),WA(1),WB(1),WC(1),V(1),R(1)
C SET WORKING VECTOR V
      DO 10 I=1,N
      V(I)=FLOAT(I)
   10 CONTINUE
C SET A UNIT MATRIX IN WB
      KK=0
      M=N*N
      DO 1 I=1,M
      WC(I)=0.0
    1 CONTINUE
      DO 2 I=1,N
      M=(I-1)*N+I
      WC(M)=1.0
    2 CONTINUE
C  MULTIPLY THE MATRIX WB BY THE GIVEN VECTOR
      CALL GMPRD (WC,V,WB,N,N,1)
      GO TO 51
```

```
C
    3 CALL GMPRD (A,WA,WB,N,N,1)
C BRANCH OUT IF HAVE REACHED MATRIX A**N TIMES V
      IF(KK-N) 51,54,54
C TRANSFER R TO THE COEFFICIENT MATRIX WC
   51 DO 52 I=1,N
      M=KK*N+I
      WC(M)=WB(I)
   52 CONTINUE
C TRANSFER WB TO WA
      DO 53 I=1,N
      WA(I)=WB(I)
   53 CONTINUE
      KK=KK+1
      GO TO 3
C PREPARE THE CONSTANT VECTOR
   54 DO 55 I=1,N
      WB(I)=-WB(I)
   55 CONTINUE
C SOLVE THE SIMULTANEOUS EQUATIONS
      CALL SIMQ (WC,WB,N,IER)
      IF(IER) 57,57,56
   56 RETURN
   57 DO 58 I=1,N
      M=N-I+1
      R(M)=WB(M)
   58 CONTINUE
      R(N+1)=1.0
      RETURN
      END
```

```
      C

C
      SUBROUTINE BART(M,XCOF,COF,COF1,ROOTR,ROOTI,CHECK,LIMI
     1,IER)
C     PURPOSE
C         TO DETERMINE ALL ROOTS OF A REAL POLYNOMIAL
C         USING BAIRSTOW METHOD
C
C     PARAMETERS
C         M       = ORDER OF POLYNOMIAL
C         XCOF    = VECTOR OF M+1 COEFFICIENTS ORDERED FROM
C                   LOW TO HIGH POWER
C         COF     = WORKING VECTOR OF LENGTH M+1
C         COF1    = WORKING VECTOR OF LENGTH M+1
C         ROOTR   = RESULTANT VECTOR OF LENGTH M CONTAINING
C                   REAL PARTS OF ROOTS OF POLYNOMIAL
C         ROOTI   = RESULTANT VECTOR OF LENGTH M CONTAINING
C                   IMAGINARY PARTS OF RROTS
C         CHECK   = VECTOR OF DIMENSION M CONTAINING LAST
C                   COEFFICIENTS OF APROXIMATING POLYNOMIAL
C         LIMI    = MAXIMUM NUMBER OF ITERATIONS
C         IER     = ERROR CODE =0 O.K.,=1 M LESS THAN 1,=2
C                   UNABLE TO DETERMINE ROOT IN LIMI
C                   ITERATI-
C                   ONS ON 10 STARTING VALUES.FOR SECOND
C                   COUPLE OF ROOTS INITIAL VALUES ARE 40
C                   =3 HIGHEST ORDER COEFFICIENT IS ZERO,=4
C                   ZERO DENOMINATOR IN KRAMER'S RULE
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C         SOLEQ
C

      DIMENSION XCOF(1),COF(1),COF1(1),ROOTR(1),ROOTI(1)
     1,CHECK(1)
      N=M
      IER=0

C ZERO SPECIAL COUNTERS

      ISC=0
      ITR=0

C SET ROOT COUNTER

      IR=1
      IRR=2
      EPSS=1.0E-07
      EPS=1.0E-06
```

```
      C
C CHECK FOR HIGHEST COEFFICIENT
      AA=XCOF(N+1)
      IF(ABS(AA)-1.0E-07) 25,25,12
C CHECK FOR DEGREE OF POLYNOMIAL
   12 IF(N-2) 14,400,45
   14 IF(N-1) 16,300,300
   16 IER=1
      RETURN
   25 IER=3
      RETURN
C SET INITIAL VALUES
   45 CO=0.0
      EO=0.0
C SET INITIAL VALUE COUNTER
   47 IN=0
   50 ICT=0
C SET C AND D TO CURRENT VALUES
      C=CO
      D=EO
      IN=IN+1
C CALCULATE B'S AND STORE THEM IN COF VECTOR
      NX=N
      NXX=N+1
      KL=N-1
      COF(NXX)=XCOF(NXX)
   51 COF(NX)=XCOF(NX)+C*COF(NXX)
      DO 56 J=1,KL
      J1=KL-J+1
      J2=J1+1
      J3=J2+1
      COF(J1)=XCOF(J1)+C*COF(J2)+D*COF(J3)
   56 CONTINUE
C CALCULATE E'S USING COF1 VECTOR
      COF1(NXX)= COF(NXX)
      COF1(NX)=COF(NX)+C*COF1(NXX)
      NY=N-2
```

```
      C
C CHECK FOR DEGREE OF POLYNOMIAL TO DETERMINE REMAINING COF1
C CHECVALUES

      IF(NY-1) 60,60,58
   58 DO 59 J=1,NY
      MT=KL-J+1
      MT1=MT+1
      MT2=MT+2
      COF1(MT)=COF(MT)+C*COF1(MT1)+D*COF1(MT2)
   59 CONTINUE
      GO TO 69
   60 COF1(KL)=COF(KL)+C*COF1(NX)+D*COF1(NXX)

C COF AND COF1 VECTORS ARE FOUND
C TRANFER LAST COEFFICIENTS

   69 B0=COF(1)
      BF=COF(2)

C CHECK FOR CONVERGENCE USING COF AND COF1 JUST FOUND
C CHECK FOR ZERO XCOF(1) AND XCOF(2)

      IF(ABS(XCOF(1))-EPS) 480,480,481
  480 IF(ABS(XCOF(2))-EPS) 482,482,484
  481 IF(ABS(XCOF(2))-EPS) 483,483,499

C START CHECKING FOR CONVERGENCE

  482 IF(ABS(B0)-EPS) 490,490,510
  490 IF(ABS(BF)-EPS) 99,99,510
  483 IF(ABS(B0)-EPSS*ABS(XCOF(1))) 487,487,510
  487 IF(ABS(BF)-EPS) 99,99,510
  484 IF(ABS(B0)-EPS) 500,500,510
  499 IF(ABS(B0)-EPSS*ABS(XCOF(1))) 500,500,510
  500 IF(ABS(BF)-EPSS*ABS(XCOF(2))) 99,99,510

C FIND DENOMINATOR USED IN KRAMER'S RULE

  510 DEN= COF1(3)*COF1(3)- COF1(2)*COF1(4)

C CHECK DENOMINATOR

      IF(ABS(DEN)-EPS) 70,70,75
   70 IF(IN-10) 71,72,72
   71 C0=C0+1.0
      E0=E0+0.5
      GO TO 50
   72 IER=4
      RETURN
```

```
      C
C COMPUTE INCREMENTS TO C AND D
   75 DC=(COF(1)*COF1(4)-COF(2)*COF1(3))/DEN
      DD=(COF(2)*COF1(2)-COF(1)*COF1(3))/DEN
C INCREMENT C AND D
      C=C+DC
      D=D+DD
C CHECK ON C,DC AND D,DD
      ICT=ICT+1
C CHECK FOR NUMBER OF ITERATIONS
      IF(ICT-LIMI) 51,80,80
C CHECK FOR INITIAL VALUE COUNTER
   80 IF(IN-10) 81,85,85
C INCREMENT INITIAL VALUES
   81 CO=CO+1.0
      EO=EO+1.0
      GO TO 50
   85 IF(ITR) 86,86,87
C SET ERROR CODE TO 2
   86 IER=2
      RETURN
   87 ISC=ISC+1
      GO TO 199
C SET LAST COEFFICIENTS INTO CHECK VECTOR
   99 CHECK(IR)=BO
      CHECK(IRR)=BF
      CALL SOLEQ(C,D,X1R,X2R,X1I,X2I)
C SET ROOTS IN VECTORS ROOTR,ROOTI
      ROOTR(IR)=X1R
      ROOTR(IRR)=X2R
      ROOTI(IR)=X1I
      ROOTI(IRR)=X2I
C STEP COUNTER FOR ROOTS
```

```
      C
      IR=IR+2
      IRR=IRR+2
C CHECK FOR ORDER OF NEW POLYNOMIAL
      IF(N-4) 149,180,190
C FIND LAST ROOT
  149 QC=-COF(NX)/COF(NXX)
      IF(ABS(QC)-EPS) 151,151,155
  151 ROOTR(IR)=0.0
      ROOTI(IR)=0.0
      RETURN
  155 ROOTR(IR)=QC
      ROOTI(IR)=0.0
      RETURN
  180 A1=-COF(NX)/COF(NXX)
      B1=-COF(KL)/COF(NXX)
      CALL SOLEQ(A1,B1,X1R,X2R,X1I,X2I)
      ROOTR(IR)=X1R
      ROOTR(IRR)=X2R
      ROOTI(IR)=X1I
      ROOTI(IRR)=X2I
      RETURN
C TRANSFER B'S VALUES INTO XCOF VECTOR TO RESTART
  190 DO 192 L=1,KL
      I2=NXX-L+1
      I1=I2-2
      XCOF(I1)=COF(I2)
  192 CONTINUE
C REDUCE DEGREE OF POLYNOMIAL AND RESTART
      N=N-2
      ITR=ITR+2
      ISC=0
  199 IF(ISC-1) 200,210,220
  200 AI=C
      BI=D
      CO=AI
      EO=BI
      GO TO 47
  210 CO=AI
      EO=-BI
      GO TO 47
  220 IF(ISC-2) 235,235,240
  235 EO=BI
```

```
      C
    CO=-AI
    GO TO 47
240 IF(ISC-3) 250,250,86
250 EO=-BI
    CO=-AI
    GO TO 47
C COMPUTE SINGLE ROOT
300 ROOTR(1)=-XCOF(1)/XCOF(2)
    RETURN
C COMPUTE  SECOND ORDER POLYNOMIAL
400 A1=-XCOF(2)/XCOF(3)
    B1=-XCOF(1)/XCOF(3)
    CALL SOLEQ(A1,B1,X1R,X2R,X1I,X2I)
    ROOTR(1)=X1R
    ROOTR(2)=X2R
    ROOTI(1)=X1I
    ROOTI(2)=X2I
    RETURN
    END
```

```
      C

C
      SUBROUTINE SOLEQ(A1,B1,X1R,X2R,X1I,X2I)

C     PURPOSE
C        TO FIND AND IDENTIFY ROOTS OF A SECOND ORDER
C        POLYNOMIAL OF THE TYPE A*X**2+B*X+C = 0
C
C     PARAMETERS
C             A1=-B/A
C             B1=-C/A
C             X1R,X2R  REAL PART OF ROOTS
C             X1I,X2I  IMAGINARY PART OF ROOTS
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C        NONE
C

      X1R=0.0
      X2R=0.0
      X1I=0.0
      X2I=0.0
      EPS1=1.0E-03

C CHECK FOR ABSOLUTE MAGNETUDE OF C AND D

      IF(ABS(A1)-1.0E-06) 8,8,10
    8 IF(ABS(B1)-1.0E-06) 30,30,12
   10 IF(ABS(B1)-1.0E-06) 13,13,14
   13 X2R=A1
      GO TO 30
   12 A1=0.0
   14 DET=A1*A1+4.0*B1
      ABD=ABS(DET)
      TEMP=SQRT(ABD)/2.0

C FIND LARGEST BETWEEN TEMP AND A1/2.0

      IF(ABS(A1/2.0)-TEMP) 40,40,50

C COMPARE THE TWO VALUES

   40 IF(ABS(A1/2.0)-EPS1*TEMP) 19,19,60
   50 IF(TEMP-EPS1*ABS(A1/2.0)) 42,42,60
   42 X1R=A1/2.0
      X2R=A1/2.0
      GO TO 30
   60 X1R=A1/2.0
      X2R=A1/2.0
   19 IF(DET) 21,21,22
   21 X1I=TEMP
```

```
      C
        X2I=-TEMP
        GO TO 30
     22 X1R=X1R+TEMP
        X2R=X2R-TEMP
     30 RETURN
        END
```

```
      C
C
      SUBROUTINE EIGCK(NOPT,L,WR,WI,W1,W2,NPR,PRR,PRI,CRIT
     1,DELT,DELTH,
     1SMAST,LUNO,L1,IER)
C        PURPOSE
C           TO CHECK FOR POSITIVE EIGENVALUES,SHIFT THEM TO
C           THE ORIGIN ( SMALLEST =0) AND DELETE THOSE THAT
C           CONTRIBUTE LESS THAN A SPECIFIED CRITERION TO THE
C           OVERALL RESPONSE
C
C        PARAMETERS
C           NOPT    = OPTION NO.5 (=0 STOP FOR +=) EI.V,=1 CO-
C                     NTINUE)
C           L       = LENGTH OF INPUT VECTORS WR,WI
C           WR      = INPUT VECTOR CONTAINING REAL PART OF
C                     EI.VS
C           WI      = INPUT VECTOR CONTAINING IMAGINARY PART
C                     OF EI.VS
C           W1,W2   = WORKING VECTORS OF LENGTH L
C           NPR     = NUMBER OF POSITIVE EI.VS
C           PRR     = VECTOR CONTAINING REAL PART OF POSITIVE
C                     EI.VS
C           PRI     = VECTOR CONTAINING IMAGINARY PART OF PO-
C                     SITIVE EI.VS
C           CRIT    = ABSOLUTE VALUE OF THE EXPONENT IN THE
C                     EXPRESSION 10**(-CRIT) USED FOR DELETING
C                     EI.VS
C
C           DELT    = TIME INTERVAL (SAMPLING TIME)
C           DELTH   = TIME SUBINTERVAL USED FOR CALCULATING
C                     MATRIX H(T)
C           SMAST   = REAL PART OF  SMALLEST (NEGATIVE) EI.V
C           LUNO    = LOGICAL UNIT NUMBER FOR CURRENT OUTPUT
C                     DEVICE
C           L1      = NEW DIMENSION OF REDUCED EI.VS
C           IER     = ERROR CODE =0 O.K.,=1 WRONG REDUCTION
C                     =2 ALLOWED POSITIVE ROOT WILL CAUSE
C                     OVER-
C                     FLOW(.GT. EXP(80)),=3 POSITIVE ROOT
C                     FOUND-PROGRAM STOPPED AS REQUESTED
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           ORDER
C
      DIMENSION WR(1),WI(1),W1(1),W2(1),PRR(1),PRI(1)

      IER=0
      L1=L
```

```
      C
C     ZERO COUNTERS
      KK=0
      NPR=0
      NNR=0
C     ZERO VECTORS PRR,PRI AND WORKING VECTORS W1,W2
      DO 5 I=1,L1
      PRR(I)=0.0
      PRI(I)=0.0
      W1(I)=0.0
      W2(I)=0.0
    5 CONTINUE
C     CHECK FOR POSITIVE REAL PART OF EIGENVALUES
      DO 21 I=1,L1
      RR=ABS(WR(I))
      IF(RR-1.E-06) 20,20,8
    8 CONTINUE
      IF(WR(I)) 20,20,10
   10 NPR=NPR+1
C     TRANSFER POSITIVE ROOTS INTO VECTOR PRR AND PRI
      PRR(NPR)=WR(I)
      PRI(NPR)=WI(I)
C     DECIDE ACCORDING TO OPTION
      IF(NOPT) 12,12,15
   12 WRITE(LUNO,13)
   13 FORMAT(1H ,/1X,'POSITIVE ROOT FOUND - SYSTEM UNSTABLE'
     1/1X,'EXECUT
     1ON STOPPED AS REQUESTED'/1X,'TO PROCEED CHANGE OP(6)
     1 TO 1'/)
      IER=3
      RETURN
   15 WRITE(LUNO,16)
   16 FORMAT(1H ,/1X,'POSITIVE ROOT FOUND - SYSTEM UNSTABLE'
     1/1X,'PROGRA
     1 IS CONTINUED AS REQUESTED'/)
      GO TO 21
   20 NNR=NNR+1
C     TRANSFER NEGATIVE ROOTS INTO W1 AND W2
      W1(NNR)=WR(I)
```

```
C     ZERO COUNTERS
      NR=0
      NPR=0
      NNR=0
C     ZERO VECTORS WPR,WPI AND WORKING VECTORS W1,W2
      DO 5 I=1,LL
      WPR(I)=0.0
      WPI(I)=0.0
      W1(I)=0.0
      W2(I)=0.0
    5 CONTINUE
C     CHECK FOR POSITIVE REAL PART OF EIGENVALUES
      DO 11 I=1,LL
      RR=REAL(WR(I))
      IF(RR-1.E-15) 20,20,8
    8 CONTINUE
      IF (WR(I)) 10,20,10
   10 NPR=NPR+1
C     TRANSFER POSITIVE ROOTS INTO VECTOR WPR AND WPI
      WPR(NPR)=WR(I)
      WPI(NPR)=WI(I)
C     DECIDE ACCORDING TO OPTION
      IF(NOPT) 13,12,15
   12 WRITE(LUNO,13)
   13 FORMAT(1H ,/1X,'POSITIVE REAL ROOT FOUND - SYSTEM UNSTABLE'
     1,1X,'EXECUT
     2ION STOPPED AS REQUESTED'/1X,'TO PROCEED CHANGE OPTIO
     3 TO 1'/)
      IERR=3
      RETURN
   15 WRITE(LUNO,16)
   16 FORMAT(1H ,/1X,'POSITIVE ROOT FOUND - SYS UNSTABLE'
     1,1X,'EXECU
     2 IS CONTINUED AS REQUESTED'/)
      GO TO 21
   20 NNR=NNR+1
C     TRANSFER NEGATIVE ROOTS INTO W1 AND W2
      W1(NNR)=WR(I)
```

```
      C
      W2(NNR)=WI(I)
   21 CONTINUE
C     CHECK FOR PROPER REDUCTION
      IF(L1-NPR-NNR) 30,31,30
   30 IER=1
      RETURN
   31 CONTINUE
C     ORDER ELEMENTS IN W1 AND W2 VECTORS
      IF(NNR-1) 33,33,32
   32 CALL ORDER(NNR,W1,W2)
   33 IF(NPR-1) 35,35,34
   34 CALL ORDER(NPR,PRR,PRI)
C     DEFINE SMALLEST NEGATIVE ELEMENT
   35 SMAST=W1(1)
C     PERFORM SHIFT TO THE ORIGIN
      DO 50 I=1,NNR
      W1(I)=W1(I)-SMAST
   50 CONTINUE
      IF(NPR-1) 70,60,60
   60 DO 61 I=1,NPR
      PRR(I)=PRR(I)-SMAST
   61 CONTINUE
C     CHECK FOR POSSIBLE OVERFLOW
      IF(PRR(NPR)*DELT-80.0) 70,70,62
   62 WRITE(6,63)
   63 FORMAT(1H0,' ALLOWED POSITIVE ROOT WILL CAUSE OVERFLOW
     1-- PROGRAM
     1 IS STOPPED AT THIS POINT'/)
      IER=2
      RETURN
C     DELETE THOSE EI.VS FOR WHICH EXP(DELTH*LAMBDA).LE.10**
C     (-CRIT)
   70 COMP=2.3026*ABS(CRIT)/DELTH
      DO 72 I=1,NNR
      RA=ABS(W1(I))
      IF(RA-COMP) 72,72,75
   72 CONTINUE
      KK=NNR
```

```
C
      GO TO 74
   75 KK=I-1
   74 CONTINUE
C     STORE ROOTS BACK INTO THE ORIGINAL VECTORS WR AND WI
      DO 76 I=1,KK
      J=KK-I+1
      WR(I)=W1(J)
      WI(I)=W2(J)
   76 CONTINUE
      IF(NPR) 100,100,80
   80 DO 85 I=1,NPR
      J=I+KK
      WR(J)=PRR(I)
      WI(J)=PRI(I)
   85 CONTINUE
      L1=KK+NPR
  100 RETURN
      END
```

```
      C

C
      SUBROUTINE ORDER(L1,A,B)

C     PURPOSE
C          TO ORDER A SET OF NUMBERS ACCORDING TO INCREA-
C          SING MAGNITUDE
C
C     PARAMETERS
C          L1     = LENGTH OF INPUT VECTORS A AND B
C          A      = INPUT VECTOR CONTAINING REAL PART OF
C                   EI.VS
C          B      = CONTAINS IMAGINARY PARTS
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          NONE
C

      DIMENSION A(1),B(1)

C SET INITIAL VALUES

      I=1
      J=2

C COMPARE TWO CONSECUTIVE VALUES

    4 IF(ABS(A(I))-ABS(A(J))) 10,10,5

C INTERCHANGE VALUES

    5 TEMP= A(I)
      A(I)=A(J)
      A(J)=TEMP

C DO THE SAME THING FOR IMAGINARY PARTS

      TEMP=B(I)
      B(I)=B(J)
      B(J)=TEMP
   10 J=J+1
      IF(J-L1) 4,4,15
   15 I=I+1
      IF(I+1-L1) 16,16,20
   16 J=I+1
      GO TO 4
   20 RETURN
      END
```

```
      C
C
      SUBROUTINE REORG(L,RR,RI,WR,WI,ND,NR,MT,NC)
C     PURPOSE
C          TO REARRANGE EIGENVALUES IN SPECIAL ORDER
C
C     PARAMETERS
C          L      = LENGTH OF VECTORS RR,RI,WR,WI
C          RR,RI  = INPUT VECTORS CONTAINING REAL AND IMAGI-
C                   NARY PART OF EI.VS RESPECTIVELY
C          WR,WI  = OUTPUT VECTORS CONTAINING REAL AND
C          IMAGI-
C                   NARY PART OF EI.VS IN ORDERED SEQUENCE
C          NR     = NUMBER OF DISTINCT EI.VS
C          NR     = NUMBER OF REPEATED EI.VS .MULTEPLICITY
C                   NUMBERS ARE STORED IN MT VECTOR
C          NC     = NUMBER OF COMPLEX EI.VS
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          NONE
C
C     NOTES
C          ORDERED SEQUENCE IS SPECIFIED BY THE VALUES OF
C          ND,NR(MT),NC
C          ND+SUM(MT(I),I=1,NR) +NC MUST BE EQUAL TO L
C
      DIMENSION RR(1),RI(1),WR(1),WI(1),MT(1)
      EPS=1.0E-06
C ZERO MULTEPLICITY VECTOR
      LT=L/2
      DO 2 I=1,LT
      MT(I)=0
    2 CONTINUE
C     SET INITIAL VALUES
      ND=0
      NR=0
      NC=0
      K=L
      I=1
C     TEST FOR COMPLEX EI.VS
    3 IF(RI(I)) 4,10,4
C     TWO COMPLEX EI.VS ARE FOUND
```

```
      C

    4 WI(K)=RI(I)
      WR(K)=RR(I)
      WI(K-1)=RI(I+1)
      WR(K-1)=RR(I+1)
C     STEP COUNTER
      NC=NC+2
C     DECIDE ACCORDING  NUMBER OF REMAINING VALUES
      IF(K-I-2) 20,7,5
C     OPERATE A SHIFT TO BASE I
    5 KN=K-2
      DO 6  J=I,KN
      RR(J)=RR(J+2)
      RI(J)=RI(J+2)
    6 CONTINUE
C     REDUCE THE ORDER OF THE SYSTEM
      K=K-2
      GO TO 3
C     SHIFT LAST VALUE
    7 RR(I)=RR(I+2)
      RI(I)=RI(I+2)
      K=K-2
      GO TO 40
   10 I=I+1
      IF(K-I) 40,40,3
   20 K=K-2
   40 IF(K-1) 45,46,50
   45 RETURN
   46 ND=ND+1
C     TRANFER LAST VALUE
      WR(1)=RR(1)
      WI(1)=RI(1)
      RETURN
   50 NN=2
      I=1
   57 J=NN
      IS=1
      COM=RR(I)
   59 IF(ABS(RR(I)-RR(J))-EPS) 60,60,70
```

```
        C

C STEP MULTEPLICITY COUNTER

   60 IS=IS+1
      WR(K)=COM

C STORE DOUBLE ROOT
C REDUCE VECTOR

      IF((J+1)-K) 61,65,66
   61 KN=K-1
      DO 63 I1=J,KN
      N1=I1+1
      RR(I1)=RR(N1)
   63 CONTINUE
      K=K-1
      GO TO 59
   65 RR(J)=RR(K)
      K=K-1
      GO TO 59
   66 K=K-1
      GO TO 72
   70 J=J+1
      IF(J-K) 59,59,71
   71 IF(IS-2) 90,72,72

C DOUBLE ROOT FOUND  **** END OF A ROW
C STEP REPEATED ROOT COUNTER

   72 NR=NR+1
      MT(NR)=IS

C TRANSFER FIRST ROOT

      WR(K)=COM
      IF(K-1) 77,77,81

C REVERSE MT VECTOR

   77 IF(NR-1) 78,78,79
   78 RETURN
   79 NR1=NR/2
      DO 80 KP=1,NR1
      KP1=NR-KP+1
      ITEM=MT(KP)
      MT(KP)=MT(KP1)
      MT(KP1)=ITEM
   80 CONTINUE
      RETURN
   81 IF(K-2) 82,82,83
```

```
      C
 82 WR(I)=RR(I+1)
    ND=ND+1
    GO TO 77
 83 KN=K-1
    DO 84 I1=I,KN
    N1=I1+1
    RR(I1)=RR(N1)
 84 CONTINUE
    K=K-1
    GO TO 57
 90 I=I+1
    NN=NN+1
    IF(NN-K) 57,57,95
 95 DO 96 KJ=1,K
    WR(KJ)=RR(KJ)
 96 CONTINUE
    ND=K
    GO TO 77
    END
```

```
      C

C
      SUBROUTINE BLDMC (L,ND,NR,MT,NC,WR,WI,WA,WB,C,IER)

C     PURPOSE
C          TO BUILD MATRIX CC USED IN DETERMINING THE FUN-
C          DAMENTAL MATRIX  ACCORDING TO THE CAYLEY-HAMILTON
C          TECHNIQUE
C
C     PARAMETERS
C          L       = LENGTH OF VECTORS WR,WI,WA,WB
C          ND      = NUMBER OF DISTINCT EI.VS
C          NR      = NUMBER OF REPEATED EI.VE
C          NR      = NUMBER OF REPEATED EI.VS
C          MT      = MULTEPLICITY VECTOR FOR REPEATED EI.VS
C          NC      = NUMBER OF COMPLEX EI.VS
C          WR,WI   = INPUT VECTORS CONTAINING REAL AND IMAGI-
C                    NARY PART OF EI.VS
C          WA,WB   = WORKING VECTORS
C          C       = OUTPUT MATRIX ARRANGED COLUMNWISE
C                    LENGTH IS L*L
C          IER     = ERROR CODE =0 O.K.,= 1 WRONG DIMENSIONS
C                    OF INPUT VECTORS
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          PDER
C

      DIMENSION MT(1),WR(1),WI(1),WA(1),WB(1),C(1)

C SET INITIAL VALUES

      KC=0
      IER =0
      LL=L*L

C CHECK IF ND+NC+SUM(MT(I),I=1,NR) = L

      ISUM=0
      IF(NR-1) 6,3,3
    3 DO 4 I=1,NR
      ISUM=ISUM+MT(I)
    4 CONTINUE
    6 KT=ND+NC+ISUM
      IF(L-KT) 7,8,7
    7 IER=1
      RETURN
    8 IF(ND) 20,20,10

C SET ND ROWS FOR DISTINCT EIGENVALUES
```

```
          10  DO 16 K=1,ND

   10  DO 16 K=1,ND
      C(K)=1.0
      K1=K+L
      C(K1)=WR(K)
      K1=K1+L
      DO 16 J=K1,LL,L
      I1=J-L
      I2=L+K
      C(J)=C(I1)*C(I2)
   16 CONTINUE
      KC=KC+ND
   20 IF(NR) 50,50,25
   25 K=KC+1
      J1=1
   29 I=1
      KP=0
C     SET FIRST ROW OF REPEATED EI.VS INTO C VECTOR AND WA
      C(K)=1.0
      WA(I)=1.0
      K1=K+L
      I=I+1
      C(K1)=WR(K)
      WA(I)=C(K1)
      K1=K1+L
      DO 28 J=K1,LL,L
      I=I+1
      I1=J-L
      I2=L+K
      C(J)=C(I1)*C(I2)
      WA(I)=C(J)
   28 CONTINUE
C     SECOND ROW OF REPEATED EI.VS CONTAINS COEFFICIENTS OF
C     OF DERIVATIVE OF PREVIOUS ROW
      K2=MT(J1)-1
      DO 40 KK=1,K2
      KP=KP+1
      I=0
C     N1 AND N2 ARE LENGTHS OF INPUT AND OUTPUT VECTORS TO
C     SUBROUTINE PDER
      N1=L-KK
      N2=N1+1
      CALL PDER (WB,N1,WA,N2)
      K=K+1
      DO 35 KS=1,KP
```

```
         10  DO 16 K=1,ND
      KJ=K+(KS-1)*L
      C(KJ)=0.0
   35 CONTINUE
      K1=KJ+L
      DO 40 J=K1,LL,L
      I=I+1
      C(J)=WB(I)
C TRANFER WB BACK INTO WA
      DO 40 KJ=1,N1
      WA(KJ)=WB(KJ)
   40 CONTINUE
      KC=K
      IF(J1-NR) 45,50,50
   45 K=K+1
      J1=J1+1
      GO TO 29
   50 IF(NC) 99,99,60
   60 K=KC+1
      ITT=2
C SET REAL PART OF COMPLEX ROOT IN KTH ROWW OF C
   69 C(K)=1.0
      K1=K+L
      C(K1)=WR(K)
C ZERO FOLLOWING ROW AND TRANFER IMAGINARY PART OF ROOT
      K2=K+1
      DO 70 J=K2,LL,L
      C(J)=0.0
   70 CONTINUE
      K3=K2+L
      C(K3)=WI(K)
C CHECK FOR DIMENSION OF MATRIX
      IF(L-2) 99,99,75
C PROVIDE REAL AND IMAGINARY PARTS OF POWERS OF COMPLEX ROOT
   75 K1=K1+L
      DO 76 J=K1,LL,L
      J1=J+1
      I1=J-L
      I2=L+K
      IC1=I1+1
      IC2=I2+1
```

```
      10  DO 16 K=1,ND

      C(J)=C(I1)*C(I2)-C(IC1)*C(IC2)
      C(J1)=C(IC1)*C(I2)+C(IC2)*C(I1)
   76 CONTINUE
      IF(NC-ITT) 99,99,79
   79 K=K+2
      ITT=ITT+2
      GO TO 69
   99 RETURN
      END
```

```
      C
C
      SUBROUTINE BLDVV (L1,WR,WI,ND,NR,MT,NC,CRIT,DELT,V,N)
C     PURPOSE
C        TO BUILD THE VECTOR OF THE EXPONENTIALS OF THE
C        EIGENVALUES AT EACH TIME SUBINTERVAL
C
C     PARAMETERS
C        L1      = LENGTH OF VECTORS WR,WI,V
C        WR,WI   = INPUT VECTORS OF REAL AND IMAGINARY PART
C                  OF EIGENVALUES RESPECTIVELY
C        ND      = NUMBER OF DISTINCT EI.VS
C        NR      = NUMBER OF REPEATED EI.VS
C        MT      = MULTEPLICITY VECTOR FOR REPEATED EI.VS
C        NC      = NUMBER OF COMPLEX EI.VS
C        CRIT    = ABSOLUTE CRITERION VALUE TO NEGLECT THE
C                  EFFECT OF SMALL EI.VS FOR LARGE TIME VA-
C                  LUES
C        DELT    = TIME VALUE AT WHICH OUTPUT VECTOR V IS
C                  CALCULATED
C        V       = OUTPUT VETOR OF EXPONENTIAL OF EI.VS
C        N       = EXTERNAL TIME INTERVAL COUNTER
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C        NONE
C
      DIMENSION WR(1),WI(1),MT(1),V(1)

C SET VECTOR V TO INITIAL VALUES OTHER THAN ZEROS

      DO 1 I=1,L1
      V(I)=1.0
    1 CONTINUE
      PR=DELT*FLOAT(N)

C SKIP CHECK FOR N=0,N=1

      IF(N-1) 25,25,10
   10 COMP=-(2.3026*CRIT)/PR

C SET TO ZERO TERMS ACCORDING TO SPECIFIED CRITERIA

      DO 20 I=1,L1
      IF(WR(I)-COMP) 16,16,20
   16 V(I)=0.0
   20 CONTINUE
   25 IF(ND) 40,40,27
   27 DO 30 I=1,ND
```

```
      C
C SKIP IF V WAS SET TO ZERO IN PREVIOUS TEST
      IF(V(I)) 28,30,28
   28 TEMP=WR(I)*PR
      V(I)=EXP(TEMP)
   30 CONTINUE
C REPEATED ROOTS
   40 J1=1
      K=ND+1
      IF(NR) 60,60,46
C CHECK FOR ZERO VALUES
   46 IF(V(K)) 47,50,47
   47 TEMP=WR(K)*PR
      V(K)=EXP(TEMP)
      K1=MT(J1)-1
      DO 48 KK=1,K1
      K2=K+KK
      V(K2)=V(K2-1)*PR
   48 CONTINUE
      J1=J1+1
      K=K+K1+1
      IF(NR-J1) 60,46,46
   50 IF(NR-J1) 60,52,51
   51 K=K+MT(J1)
      J1=J1+1
      GO TO 46
   52 K=K+MT(J1)
   60 IF(NC) 80,80,65
   65 ITT=2
   66 IF(V(K)) 70,71,70
   70 TEMP=WR(K)*PR
      TEMP1=WI(K)*PR
      V(K)=EXP(TEMP)*COS(TEMP1)
      V(K+1)=EXP(TEMP)*SIN(TEMP1)
   71 IF(NC-ITT) 80,80,72
   72 K=K+2
      ITT=ITT+2
      GO TO 66
   80 RETURN
      END
```

```
      C
C
      SUBROUTINE POSRT(N,A,FM,WORK1,WORK7,NIT,DELTH,K1,IER)

C     PURPOSE
C          TO CALCULATE THE FUNDAMENTAL MATRIX BY THE POWER
C          SERIES METHOD
C
C     PARAMETERS
C          N=DIMENSION OF MATRIX A,FM,WORK1,WORK7
C          A=COEFFICIENT MATRIX OF STATE EQUATION
C          FM=FUNDAMENTAL MATRIX
C          WORK1,WORK7=WORKING VECTORS OF LENGTH N*N
C          NIT=NUMBER OF ITERATIONS
C          DELTH=TIME SUBINTERVAL
C          K1=TIME SUBINTERVAL COUNTER
C          IER=ERROR CODE = 0 O.K., = 1 NO CONVERGENCE IN
C          NIT ITERATIONS
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          GMPRD,GMADD,MSMUL
C

      DIMENSION A(1),FM(1),WORK1(1),WORK7(1)

      IER=0
      EPS=1.0E-03
      IACC=1
      NN=N*N
      TEE=DELTH*FLOAT(K1)

C     SET UP UNIT MATRICES IN FM AND WORK1
C

      DO 4 I=1,NN
      FM(I)=0.0
      WORK1(I)=0.0
    4 CONTINUE
      DO 5 I=1,N
      K=I+N*(I-1)
      FM(K)=1.0
      WORK1(K)=1.0
    5 CONTINUE

C     BYPASS THE SUBROUTINE THE FIRST TIME THROUGH.
C

      IF(K1) 100,100,6
    6 CONTINUE
```

```
      C
C     PRODUCE SUCCESSIVE POWERS OF A,MULTIPLY THEM BY THE
C     TIME FACTOR
C     AND ADD THEM TOGETHER AFTER HAVING CHECKED FOR
C     CONVERGENCE
C
      DO 80 I=1,NIT
      IACC=IACC*I
      TEEF=TEE**I/FLOAT(IACC)
      CALL GMPRD(WORK1,A,WORK7,N,N,N)
C     TRANSFER WORK7 TO WORK1
C
      DO 10 K=1,NN
      WORK1(K)=WORK7(K)
   10 CONTINUE
      CALL MSMUL(N,N,WORK7,TEEF,WORK7)
C     CHECK FOR CONVERGENCE
C
      ISEE=0
      DO 30 K=1,NN
      IF(ABS(WORK7(K))-EPS*ABS(FM(K))) 21,21,30
   21 ISEE=ISEE+1
   30 CONTINUE
      IF(ISEE-NN) 40,100,100
   40 CALL GMADD(FM,WORK7,FM,N,N)
   80 CONTINUE
      IER=1
  100 RETURN
      END
```

```
      C
C
      SUBROUTINE BLDMR (N,A,WA,R,KK)
C     PURPOSE
C        TO BUILD SUCCESSIVE POWERS OF COEFFICIENT MATRIX
C        A
C
C     PARAMETERS
C        N      = DIMENSION OF INPUT MATRIX A .LENGTH IS
C                 N*N
C        A      = INPUT MATRIX
C        WA     = WORKING VECTOR OF LENGTH N*N
C        R      = RESULTANT MATRIX EXPRESSED COLUMNWISE
C                 OF LENGTH N*N
C        KK     = EXTERNAL COUNTER .IT VARIES BETWEEN 0
C                 AND L.CONSEQUENTLY MAXIMUM POWER OF A
C                 CALCULATED IS L-1
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C        GMPRD
C
      DIMENSION A(1),WA(1),R(1)
      M=N*N
       IF(KK-2) 10,20,30
C SET A UNIT MATRIX IN R
   10 DO 11 I=1,M
      R(I)=0.0
   11 CONTINUE
      DO 12 I=1,N
      I1=(I-1)*N+I
      R(I1)=1.0
   12 CONTINUE
      GO TO 50
   20 DO 25 I=1,M
      R(I)=A(I)
   25 CONTINUE
      GO TO 50
   30 CALL GMPRD (WA,A,R,N,N,N)
C TRANSFER LAST MATRIX IN WA
      DO 35 I=1,M
      WA(I)=R(I)
   35 CONTINUE
   50 RETURN
      END
```

```
      C
C
      SUBROUTINE MSMUL (L,L1,A,S,R)
C     PURPOSE
C          TO MULTIPLY A MATRIX BY A CONSTANT
C
C     PARAMETERS
C          L      = NUMBER OF ROWS OF INPUT MATRIX A
C          L1     = NUMBER OF COLUMNS OF MATRIX A
C          A      = INPUT MATRIX ARRANGED COLUMNWISE
C          S      = CONSTANT
C          R      = OUTPUT MATRIX ARRANGED COLUMNWISE.IT
C                   COULD BE THE SAME AS A
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          NONE
C
      DIMENSION A(1),R(1)
      LL=L*L1
      DO 1 I=1,LL
      R(I)=A(I)*S
    1 CONTINUE
      RETURN
      END
```

```
      C
C
      SUBROUTINE BLDFF(NTD,TO,DELT,X,N2,NU,UD,OV,NP,MV,NCODE
     1,WA,WB,MAXO
     1,OW,ISAVE,IOWCT,N,LUNP)
C     PURPOSE
C        TO BUILD THE FORCING FUNCTION VECTORS U AND V AT
C        EACH TIME INTERVAL
C
C     PARAMETERS
C        NTD     = NUMBER OF TIME DELAYS
C        TO      = INITIAL TIME VALUE
C        DELT    = TIME INTERVAL
C        X       = STATE VARIABLE VECTOR
C        N2      = LENGTH OF VECTORS NU,UD,OV
C        NU      = VECTOR CONTAINING INDECES OF VARIABLES
C                  WHOSE NUMBER IS SPECIFIED IN VECTOR MV
C        UD      = VECTOR CONTAINING VALUES OF TIME DELAYS
C                  ASSOCIATED WITH VARIABLES OF NU VECTOR
C        NP      = LENGTH OF VECTOR MV
C        MV      = VECTOR CONTAINING NUMBERS OF EXTERNAL
C                  FORCING FUNCTIONS AND AS MANY COUPLES OF
C                  NUMBERS OF DELAYED FORCING FUNCTIONS AND
C                  STATE VARIABLES AS THERE ARE TIME DELAYS
C        NCODE   = VECTOR  OF LENGTH  NF= NUMBER OF EXTER-
C                  NAL FORCING FUNCTIONS CONTAINING CODE
C                  NU-
C                  MBERS FOR THE TYPE OF DISTURBACES
C        WA      = VCTOR OF LENGTH NF CONTAINING THE FIRST
C                  DISTURBANCE PARAMETER
C        WB      = VECTOR OF LENGTH NF CONTAINING THE SE-
C                  COND DISTURBANCE PARAMETER
C        MAXOW   = LENGTH OF VECTOR OW
C        OW      = VECTOR FOR TEMPORARY STORAGE OF DELAYED
C                  FORCING FUNCTIONSAND STATE VARIABLES
C        ISAVE   = WORKING VECTOR CONTAINING LAST POSITIONS
C                  OF DELAYED VARIABLES WHEN OW IS REUSED
C        IOWCT   = COUNTER INITIALIZED TO ZERO OUTSIDE THE
C                   ITERATION LOOP
C        N       = TIME INTERVAL COUNTER
C        LUNP    = LOGICAL UNIT NUMBER FOR PRINTING  OUT AN
C                  ERROR MESSAGE
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C        NONE
C
      DIMENSION X(1),NU(1),UD(1),OV(1),MV(1),NCODE(1),WA(1)
     1,WB(1),OW(1)
     1ISAVE(1)
```

```
      C

C     SET INITIAL VALUES.

      K=1
      J=1
      I=1
      NUXD=0
      IFLAG=-1

      NI=TO/DELT+.5
      NCT=N+NI
      T=DELT*FLOAT(NCT)

      KV=MV(K)
      IF(KV) 5,5,7
    5 I=I-1
      GO TO 110

    7 CONTINUE

C     START GENERATING DISTURBANCES ACCORDING TO CODE.

      KK=NU(I)
      KS=NCODE(KK)
      A1=WA(KK)
      B1=WB(KK)
      GO TO(10,20,30,40,50,60,70,80,90,100),KS
   10 TEMP=A1
      GO TO 105
   20 TEMP=A1*T
      GO TO 105
   30 TEMP=A1*SIN(B1*T)
      GO TO 105
   40 IF(B1-T) 42,41,41
   41 TEMP=A1
      GO TO 105
   42 TEMP=0.0
      GO TO 105
   50 WRITE(LUNP,51)
   51 FORMAT(1H0,'FORCING FUNCTION REQUESTED NOT AVAILABLE'
     1/)
      RETURN
   60 GO TO 50
   70 GO TO 50
   80 GO TO 50
   90 GO TO 50
  100 GO TO 50

  105 CONTINUE
      IF(K-1) 107,107,130
```

```
      C
 107 OV(I)=TEMP
     KV=KV-1
     IF(KV) 110,110,108
 108 I=I+1
     GO TO 7
 110 CONTINUE
     IF(NTD) 600,600,120
 120 DO 122 KR=2,NP
     NUXD=NUXD+MV(KR)
 122 CONTINUE
C
C    CHECK FOR SPACE IN VECTOR OW.
C
     DO 125 KR=1,N2
     KK=ABS(UD(KR))/DELT+0.5
     IF(KK*NUXD-MAXOW) 125,125,126
 125 CONTINUE
     GO TO 128
 126 WRITE(LUNP,127)
 127 FORMAT(1H ,'TOO MANY VALUES IN VECTOR OW-- REDUCE NO.
    1 OF POINTS'/
     RETURN
 128 NSETS=MAXOW/NUXD

     K=K+1
     I=I+1
     KV=MV(K)
     IF(KV) 129,129,7
 129 I=I-1
     GO TO 300

 130 CONTINUE
     NSUB=NCT-NSETS*IOWCT
     JT=NSETS*NUXD
     JC=J+NSUB*NUXD
     IF(JC-JT) 150,150,140
 140 IOWCT=IOWCT+1
     GO TO 130
 150 OW(JC)=TEMP
     IDLAY=ABS(UD(I))/DELT+0.5
     IF(IOWCT) 170,170,250
 170 CONTINUE
     IF(JC-J-IDLAY*NUXD) 180,200,200
 180 OV(I)=0.0
     GO TO 280
 200 ISAVE(J)= JC-IDLAY*NUXD
     KR=ISAVE(J)
     OV(I)=OW(KR)
     GO TO 280
 250 KR=ISAVE(J)+NUXD
```

```
      C
      IF(KR-JT) 260,260,270
  260 ISAVE(J)=KR
      OV(I)=OW(KR)
      GO TO 280
  270 ISAVE(J)=KR-JT
      KR=KR-JT
      OV(I)=OW(KR)

  280 KV=KV-1
      IF(KV) 300,300,290
  290 I=I+1
      J=J+1
      IF(IFLAG) 7,7,330
  300 K=K+1
      I=I+1
      IF(K-NP) 400,320,600
  320 KV=MV(K)
      IF(KV) 600,600,325
  325 J=J+1
  330 KK=NU(I)
      TEMP=X(KK)
      IFLAG=1
      GO TO 130
  400 KV=MV(K)
      GO TO (7,130,430,450,430,450,430,450),K
  430 IF(KV) 440,440,325
  440 K=K+1
      GO TO 400
  450 IF(KV) 460,460,470
  460 K=K+1
      GO TO 400
  470 IFLAG=-1
      J=J+1
      GO TO 7
  600 RETURN
      END
```

```
      C

C
      SUBROUTINE AFCRT(N,INPUT,L,IOUT)

C     PURPOSE
C        TO CONVERT ALPHAMERICS FROM A1 TO A2 FORMAT
C
C     PARAMETERS
C        N      = LENGTH OF INPUT VECTOR INPUT (A1)
C        INPUT  = INPUT VECTOR
C        L      = LENGTH OF OUTPUT VECTOR IOUT (A2) = N/2
C        IOUT   = OUTPUT VECTOR
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C        NONE
C

      DIMENSION INPUT(1),IOUT(1)
      DATA MSK,MSK1,MSK2/ZFF00,Z8000,Z7F00/
      L=N/2+.5
      DO 10 I=1,L
      K=1+2*(I-1)
      K1=K+1
      J=INPUT(K)
      J1=INPUT(K1)
      IOUT(I)=IAND(MSK,J)+IAND(MSK2,J1)/256
      IR=IAND(MSK1,J1)
      IF(IR)2,10,10
    2 IOUT(I)=IOUT(I)+128
   10 CONTINUE
      RETURN
      END
```